



CryptoAuthLib

v3.2.2

<b>1 License</b>	<b>1</b>
<b>2 calib directory - Purpose</b>	<b>3</b>
<b>3 crypto directory - Purpose</b>	<b>5</b>
<b>4 HAL Directory - Purpose</b>	<b>7</b>
<b>5 mbedtls directory - Purpose</b>	<b>9</b>
<b>6 openssl directory - Purpose</b>	<b>11</b>
<b>7 IP Protection with Symmetric Authentication</b>	<b>13</b>
<b>8 Setting up cryptoauthlib as a PKCS11 Provider for your system (LINUX)</b>	<b>15</b>
<b>9 app directory - Purpose</b>	<b>21</b>
<b>10 Secure boot using ATECC608</b>	<b>23</b>
<b>11 TNG Functions</b>	<b>25</b>
<b>12 CryptoAuthLib - Microchip CryptoAuthentication Library</b>	<b>27</b>
<b>13 Deprecated List</b>	<b>33</b>
<b>14 Todo List</b>	<b>35</b>
<b>15 Module Index</b>	<b>37</b>
15.1 Modules . . . . .	37
<b>16 Data Structure Index</b>	<b>39</b>
16.1 Data Structures . . . . .	39
<b>17 File Index</b>	<b>43</b>
17.1 File List . . . . .	43
<b>18 Module Documentation</b>	<b>51</b>
18.1 Basic Crypto API methods (atcab_) . . . . .	51
18.1.1 Detailed Description . . . . .	59
18.1.2 Macro Definition Documentation . . . . .	59
18.1.2.1 ATCA_AES_GCM_IV_STD_LENGTH . . . . .	59
18.1.2.2 atca_execute_command . . . . .	59
18.1.2.3 atcab_cfg_discover . . . . .	59
18.1.2.4 atcab_get_addr . . . . .	59
18.1.2.5 SHA_CONTEXT_MAX_SIZE . . . . .	59
18.1.3 Typedef Documentation . . . . .	59
18.1.3.1 atca_aes_gcm_ctx_t . . . . .	60
18.1.4 Function Documentation . . . . .	60

18.1.4.1 _atcab_exit()	60
18.1.4.2 atcab_aes()	60
18.1.4.3 atcab_aes_cbc_decrypt_block()	60
18.1.4.4 atcab_aes_cbc_encrypt_block()	61
18.1.4.5 atcab_aes_cbc_encrypt_block_ext()	61
18.1.4.6 atcab_aes_cbc_init()	61
18.1.4.7 atcab_aes_cbc_init_ext()	62
18.1.4.8 atcab_aes_cmac_finish()	62
18.1.4.9 atcab_aes_cmac_init()	63
18.1.4.10 atcab_aes_cmac_init_ext()	63
18.1.4.11 atcab_aes_cmac_update()	64
18.1.4.12 atcab_aes_ctr_block()	64
18.1.4.13 atcab_aes_ctr_decrypt_block()	65
18.1.4.14 atcab_aes_ctr_encrypt_block()	65
18.1.4.15 atcab_aes_ctr_increment()	65
18.1.4.16 atcab_aes_ctr_init()	66
18.1.4.17 atcab_aes_ctr_init_ext()	66
18.1.4.18 atcab_aes_ctr_init_rand()	67
18.1.4.19 atcab_aes_ctr_init_rand_ext()	67
18.1.4.20 atcab_aes_decrypt()	68
18.1.4.21 atcab_aes_decrypt_ext()	69
18.1.4.22 atcab_aes_encrypt()	69
18.1.4.23 atcab_aes_encrypt_ext()	70
18.1.4.24 atcab_aes_gcm_aad_update()	70
18.1.4.25 atcab_aes_gcm_decrypt_finish()	71
18.1.4.26 atcab_aes_gcm_decrypt_update()	71
18.1.4.27 atcab_aes_gcm_encrypt_finish()	71
18.1.4.28 atcab_aes_gcm_encrypt_update()	72
18.1.4.29 atcab_aes_gcm_init()	72
18.1.4.30 atcab_aes_gcm_init_rand()	73
18.1.4.31 atcab_aes_gfm()	73
18.1.4.32 atcab_challenge()	74
18.1.4.33 atcab_challenge_seed_update()	74
18.1.4.34 atcab_checkmac()	75
18.1.4.35 atcab_cmp_config_zone()	75
18.1.4.36 atcab_counter()	76
18.1.4.37 atcab_counter_increment()	76
18.1.4.38 atcab_counter_read()	76
18.1.4.39 atcab_derivekey()	77
18.1.4.40 atcab_ecdh()	77
18.1.4.41 atcab_ecdh_base()	78
18.1.4.42 atcab_ecdh_enc()	78

18.1.4.43 atcab_ecdh_ioenc()	79
18.1.4.44 atcab_ecdh_tempkey()	79
18.1.4.45 atcab_ecdh_tempkey_ioenc()	80
18.1.4.46 atcab_gendig()	80
18.1.4.47 atcab_genkey()	81
18.1.4.48 atcab_genkey_base()	81
18.1.4.49 atcab_get_device()	82
18.1.4.50 atcab_get_device_type()	82
18.1.4.51 atcab_get_device_type_ext()	82
18.1.4.52 atcab_get_pubkey()	82
18.1.4.53 atcab_get_zone_size()	83
18.1.4.54 atcab_hmac()	83
18.1.4.55 atcab_hw_sha2_256()	84
18.1.4.56 atcab_hw_sha2_256_finish()	84
18.1.4.57 atcab_hw_sha2_256_init()	84
18.1.4.58 atcab_hw_sha2_256_update()	85
18.1.4.59 atcab_idle()	85
18.1.4.60 atcab_info()	85
18.1.4.61 atcab_info_base()	86
18.1.4.62 atcab_info_get_latch()	86
18.1.4.63 atcab_info_set_latch()	87
18.1.4.64 atcab_init()	87
18.1.4.65 atcab_init_device()	87
18.1.4.66 atcab_init_ext()	88
18.1.4.67 atcab_is_ca_device()	88
18.1.4.68 atcab_is_config_locked()	88
18.1.4.69 atcab_is_data_locked()	89
18.1.4.70 atcab_is_locked()	89
18.1.4.71 atcab_is_slot_locked()	89
18.1.4.72 atcab_is_ta_device()	90
18.1.4.73 atcab_kdf()	90
18.1.4.74 atcab_lock()	91
18.1.4.75 atcab_lock_config_zone()	91
18.1.4.76 atcab_lock_config_zone_crc()	91
18.1.4.77 atcab_lock_data_slot()	92
18.1.4.78 atcab_lock_data_zone()	92
18.1.4.79 atcab_lock_data_zone_crc()	92
18.1.4.80 atcab_mac()	93
18.1.4.81 atcab_nonce()	93
18.1.4.82 atcab_nonce_base()	94
18.1.4.83 atcab_nonce_load()	94
18.1.4.84 atcab_nonce_rand()	95

18.1.4.85 atcab_printbin()	95
18.1.4.86 atcab_priv_write()	95
18.1.4.87 atcab_random()	96
18.1.4.88 atcab_random_ext()	96
18.1.4.89 atcab_read_bytes_zone()	96
18.1.4.90 atcab_read_config_zone()	97
18.1.4.91 atcab_read_enc()	97
18.1.4.92 atcab_read_pubkey()	98
18.1.4.93 atcab_read_serial_number()	98
18.1.4.94 atcab_read_sig()	99
18.1.4.95 atcab_read_zone()	99
18.1.4.96 atcab_release()	100
18.1.4.97 atcab_release_ext()	100
18.1.4.98 atcab_secureboot()	100
18.1.4.99 atcab_secureboot_mac()	101
18.1.4.100 atcab_selftest()	101
18.1.4.101 atcab_sha()	102
18.1.4.102 atcab_sha_base()	102
18.1.4.103 atcab_sha_end()	103
18.1.4.104 atcab_sha_hmac()	103
18.1.4.105 atcab_sha_hmac_finish()	104
18.1.4.106 atcab_sha_hmac_init()	104
18.1.4.107 atcab_sha_hmac_update()	105
18.1.4.108 atcab_sha_read_context()	105
18.1.4.109 atcab_sha_start()	105
18.1.4.110 atcab_sha_update()	106
18.1.4.111 atcab_sha_write_context()	106
18.1.4.112 atcab_sign()	106
18.1.4.113 atcab_sign_base()	107
18.1.4.114 atcab_sign_internal()	107
18.1.4.115 atcab_sleep()	108
18.1.4.116 atcab_updateextra()	108
18.1.4.117 atcab_verify()	108
18.1.4.118 atcab_verify_extern()	109
18.1.4.119 atcab_verify_extern_mac()	110
18.1.4.120 atcab_verify_invalidate()	110
18.1.4.121 atcab_verify_stored()	111
18.1.4.122 atcab_verify_stored_mac()	111
18.1.4.123 atcab_verify_validate()	112
18.1.4.124 atcab_version()	112
18.1.4.125 atcab_wakeup()	113
18.1.4.126 atcab_write()	113

18.1.4.127 atcab_write_bytes_zone()	113
18.1.4.128 atcab_write_config_counter()	114
18.1.4.129 atcab_write_config_zone()	114
18.1.4.130 atcab_write_enc()	115
18.1.4.131 atcab_write_pubkey()	115
18.1.4.132 atcab_write_zone()	116
18.1.4.133 calib_aes_gcm_aad_update()	116
18.1.4.134 calib_aes_gcm_decrypt_finish()	117
18.1.4.135 calib_aes_gcm_decrypt_update()	117
18.1.4.136 calib_aes_gcm_encrypt_finish()	118
18.1.4.137 calib_aes_gcm_encrypt_update()	118
18.1.4.138 calib_aes_gcm_init()	119
18.1.4.139 calib_aes_gcm_init_rand()	119
18.1.5 Variable Documentation	120
18.1.5.1 _gDevice	120
18.1.5.2 atca_basic_aes_gcm_version	120
18.2 Configuration (cfg_)	121
18.3 ATCACommand (atca_)	122
18.3.1 Detailed Description	122
18.3.2 Typedef Documentation	122
18.3.2.1 ATCACommand	122
18.3.3 Function Documentation	122
18.3.3.1 deleteATCACommand()	122
18.3.3.2 initATCACommand()	123
18.3.3.3 newATCACommand()	123
18.4 ATCADevice (atca_)	124
18.4.1 Detailed Description	127
18.4.2 Macro Definition Documentation	127
18.4.2.1 ATCA_AES_ENABLE_EN_MASK	127
18.4.2.2 ATCA_AES_ENABLE_EN_SHIFT	127
18.4.2.3 ATCA_CHIP_MODE_CLK_DIV	127
18.4.2.4 ATCA_CHIP_MODE_CLK_DIV_MASK	127
18.4.2.5 ATCA_CHIP_MODE_CLK_DIV_SHIFT	127
18.4.2.6 ATCA_CHIP_MODE_I2C_EXTRA_MASK	127
18.4.2.7 ATCA_CHIP_MODE_I2C_EXTRA_SHIFT	128
18.4.2.8 ATCA_CHIP_MODE_TTL_EN_MASK	128
18.4.2.9 ATCA_CHIP_MODE_TTL_EN_SHIFT	128
18.4.2.10 ATCA_CHIP_MODE_WDG_LONG_MASK	128
18.4.2.11 ATCA_CHIP_MODE_WDG_LONG_SHIFT	128
18.4.2.12 ATCA_CHIP_OPT_ECDH_PROT	128
18.4.2.13 ATCA_CHIP_OPT_ECDH_PROT_MASK	128
18.4.2.14 ATCA_CHIP_OPT_ECDH_PROT_SHIFT	129

18.4.2.15 ATCA_CHIP_OPT_IO_PROT_EN_MASK . . . . .	129
18.4.2.16 ATCA_CHIP_OPT_IO_PROT_EN_SHIFT . . . . .	129
18.4.2.17 ATCA_CHIP_OPT_IO_PROT_KEY . . . . .	129
18.4.2.18 ATCA_CHIP_OPT_IO_PROT_KEY_MASK . . . . .	129
18.4.2.19 ATCA_CHIP_OPT_IO_PROT_KEY_SHIFT . . . . .	129
18.4.2.20 ATCA_CHIP_OPT_KDF_AES_EN_MASK . . . . .	129
18.4.2.21 ATCA_CHIP_OPT_KDF_AES_EN_SHIFT . . . . .	130
18.4.2.22 ATCA_CHIP_OPT_KDF_PROT . . . . .	130
18.4.2.23 ATCA_CHIP_OPT_KDF_PROT_MASK . . . . .	130
18.4.2.24 ATCA_CHIP_OPT_KDF_PROT_SHIFT . . . . .	130
18.4.2.25 ATCA_CHIP_OPT_POST_EN_MASK . . . . .	130
18.4.2.26 ATCA_CHIP_OPT_POST_EN_SHIFT . . . . .	130
18.4.2.27 ATCA_COUNTER_MATCH_EN_MASK . . . . .	130
18.4.2.28 ATCA_COUNTER_MATCH_EN_SHIFT . . . . .	131
18.4.2.29 ATCA_COUNTER_MATCH_KEY . . . . .	131
18.4.2.30 ATCA_COUNTER_MATCH_KEY_MASK . . . . .	131
18.4.2.31 ATCA_COUNTER_MATCH_KEY_SHIFT . . . . .	131
18.4.2.32 ATCA_I2C_ENABLE_EN_MASK . . . . .	131
18.4.2.33 ATCA_I2C_ENABLE_EN_SHIFT . . . . .	131
18.4.2.34 ATCA_KEY_CONFIG_AUTH_KEY . . . . .	131
18.4.2.35 ATCA_KEY_CONFIG_AUTH_KEY_MASK . . . . .	132
18.4.2.36 ATCA_KEY_CONFIG_AUTH_KEY_SHIFT . . . . .	132
18.4.2.37 ATCA_KEY_CONFIG_KEY_TYPE . . . . .	132
18.4.2.38 ATCA_KEY_CONFIG_KEY_TYPE_MASK . . . . .	132
18.4.2.39 ATCA_KEY_CONFIG_KEY_TYPE_SHIFT . . . . .	132
18.4.2.40 ATCA_KEY_CONFIG_LOCKABLE_MASK . . . . .	132
18.4.2.41 ATCA_KEY_CONFIG_LOCKABLE_SHIFT . . . . .	132
18.4.2.42 ATCA_KEY_CONFIG_PERSIST_DISABLE_MASK . . . . .	133
18.4.2.43 ATCA_KEY_CONFIG_PERSIST_DISABLE_SHIFT . . . . .	133
18.4.2.44 ATCA_KEY_CONFIG_PRIVATE_MASK . . . . .	133
18.4.2.45 ATCA_KEY_CONFIG_PRIVATE_SHIFT . . . . .	133
18.4.2.46 ATCA_KEY_CONFIG_PUB_INFO_MASK . . . . .	133
18.4.2.47 ATCA_KEY_CONFIG_PUB_INFO_SHIFT . . . . .	133
18.4.2.48 ATCA_KEY_CONFIG_REQ_AUTH_MASK . . . . .	133
18.4.2.49 ATCA_KEY_CONFIG_REQ_AUTH_SHIFT . . . . .	133
18.4.2.50 ATCA_KEY_CONFIG_REQ_RANDOM_MASK . . . . .	134
18.4.2.51 ATCA_KEY_CONFIG_REQ_RANDOM_SHIFT . . . . .	134
18.4.2.52 ATCA_KEY_CONFIG_RFU_MASK . . . . .	134
18.4.2.53 ATCA_KEY_CONFIG_RFU_SHIFT . . . . .	134
18.4.2.54 ATCA_KEY_CONFIG_X509_ID . . . . .	134
18.4.2.55 ATCA_KEY_CONFIG_X509_ID_MASK . . . . .	134
18.4.2.56 ATCA_KEY_CONFIG_X509_ID_SHIFT . . . . .	134

18.4.2.57 ATCA_PACKED	135
18.4.2.58 ATCA_SECURE_BOOT_DIGEST	135
18.4.2.59 ATCA_SECURE_BOOT_DIGEST_MASK	135
18.4.2.60 ATCA_SECURE_BOOT_DIGEST_SHIFT	135
18.4.2.61 ATCA_SECURE_BOOT_MODE	135
18.4.2.62 ATCA_SECURE_BOOT_MODE_MASK	135
18.4.2.63 ATCA_SECURE_BOOT_MODE_SHIFT	135
18.4.2.64 ATCA_SECURE_BOOT_PERSIST_EN_MASK	136
18.4.2.65 ATCA_SECURE_BOOT_PERSIST_EN_SHIFT	136
18.4.2.66 ATCA_SECURE_BOOT_PUB_KEY	136
18.4.2.67 ATCA_SECURE_BOOT_PUB_KEY_MASK	136
18.4.2.68 ATCA_SECURE_BOOT_PUB_KEY_SHIFT	136
18.4.2.69 ATCA_SECURE_BOOT_RAND_NONCE_MASK	136
18.4.2.70 ATCA_SECURE_BOOT_RAND_NONCE_SHIFT	136
18.4.2.71 ATCA_SLOT_CONFIG_ECDH_MASK	137
18.4.2.72 ATCA_SLOT_CONFIG_ECDH_SHIFT	137
18.4.2.73 ATCA_SLOT_CONFIG_ENCRYPTED_READ_MASK	137
18.4.2.74 ATCA_SLOT_CONFIG_ENCRYPTED_READ_SHIFT	137
18.4.2.75 ATCA_SLOT_CONFIG_EXT_SIG_MASK	137
18.4.2.76 ATCA_SLOT_CONFIG_EXT_SIG_SHIFT	137
18.4.2.77 ATCA_SLOT_CONFIG_GEN_KEY_MASK	137
18.4.2.78 ATCA_SLOT_CONFIG_GEN_KEY_SHIFT	137
18.4.2.79 ATCA_SLOT_CONFIG_INT_SIG_MASK	138
18.4.2.80 ATCA_SLOT_CONFIG_INT_SIG_SHIFT	138
18.4.2.81 ATCA_SLOT_CONFIG_IS_SECRET_MASK	138
18.4.2.82 ATCA_SLOT_CONFIG_IS_SECRET_SHIFT	138
18.4.2.83 ATCA_SLOT_CONFIG_LIMITED_USE_MASK	138
18.4.2.84 ATCA_SLOT_CONFIG_LIMITED_USE_SHIFT	138
18.4.2.85 ATCA_SLOT_CONFIG_NOMAC_MASK	138
18.4.2.86 ATCA_SLOT_CONFIG_NOMAC_SHIFT	138
18.4.2.87 ATCA_SLOT_CONFIG_PRIV_WRITE_MASK	139
18.4.2.88 ATCA_SLOT_CONFIG_PRIV_WRITE_SHIFT	139
18.4.2.89 ATCA_SLOT_CONFIG_READKEY	139
18.4.2.90 ATCA_SLOT_CONFIG_READKEY_MASK	139
18.4.2.91 ATCA_SLOT_CONFIG_READKEY_SHIFT	139
18.4.2.92 ATCA_SLOT_CONFIG_WRITE_CONFIG	139
18.4.2.93 ATCA_SLOT_CONFIG_WRITE_CONFIG_MASK	139
18.4.2.94 ATCA_SLOT_CONFIG_WRITE_CONFIG_SHIFT	140
18.4.2.95 ATCA_SLOT_CONFIG_WRITE_ECDH_MASK	140
18.4.2.96 ATCA_SLOT_CONFIG_WRITE_ECDH_SHIFT	140
18.4.2.97 ATCA_SLOT_CONFIG_WRITE_KEY	140
18.4.2.98 ATCA_SLOT_CONFIG_WRITE_KEY_MASK	140



18.4.2.99 ATCA_SLOT_CONFIG_WRITE_KEY_SHIFT . . . . .	140
18.4.2.100 ATCA_SLOT_LOCKED . . . . .	140
18.4.2.101 ATCA_USE_LOCK_ENABLE_MASK . . . . .	141
18.4.2.102 ATCA_USE_LOCK_ENABLE_SHIFT . . . . .	141
18.4.2.103 ATCA_USE_LOCK_KEY_MASK . . . . .	141
18.4.2.104 ATCA_USE_LOCK_KEY_SHIFT . . . . .	141
18.4.2.105 ATCA_VOL_KEY_PERM_EN_MASK . . . . .	141
18.4.2.106 ATCA_VOL_KEY_PERM_EN_SHIFT . . . . .	141
18.4.2.107 ATCA_VOL_KEY_PERM_SLOT . . . . .	141
18.4.2.108 ATCA_VOL_KEY_PERM_SLOT_MASK . . . . .	142
18.4.2.109 ATCA_VOL_KEY_PERM_SLOT_SHIFT . . . . .	142
18.4.3 Typedef Documentation . . . . .	142
18.4.3.1 ATCADevice . . . . .	142
18.4.3.2 atecc508a_config_t . . . . .	142
18.4.3.3 atecc608_config_t . . . . .	142
18.4.3.4 atsha204a_config_t . . . . .	142
18.4.4 Enumeration Type Documentation . . . . .	142
18.4.4.1 ATCADeviceType . . . . .	142
18.4.5 Function Documentation . . . . .	143
18.4.5.1 atGetCommands() . . . . .	143
18.4.5.2 atGetIFace() . . . . .	143
18.4.5.3 deleteATCADevice() . . . . .	144
18.4.5.4 initATCADevice() . . . . .	144
18.4.5.5 newATCADevice() . . . . .	144
18.4.5.6 releaseATCADevice() . . . . .	145
18.5 ATCAIface (atca_) . . . . .	146
18.5.1 Detailed Description . . . . .	147
18.5.2 Typedef Documentation . . . . .	147
18.5.2.1 ATCAIface . . . . .	147
18.5.3 Enumeration Type Documentation . . . . .	147
18.5.3.1 ATCAIfaceType . . . . .	147
18.5.3.2 ATCAKitType . . . . .	147
18.5.4 Function Documentation . . . . .	148
18.5.4.1 atgetifacecfg() . . . . .	148
18.5.4.2 atgetifacehaldat() . . . . .	148
18.5.4.3 atidle() . . . . .	148
18.5.4.4 atinit() . . . . .	149
18.5.4.5 atpostinit() . . . . .	149
18.5.4.6 atreceive() . . . . .	149
18.5.4.7 atsend() . . . . .	150
18.5.4.8 atsleep() . . . . .	150
18.5.4.9 atwake() . . . . .	151

18.5.4.10 deleteATCAIface()	151
18.5.4.11 initATCAIface()	151
18.5.4.12 newATCAIface()	152
18.5.4.13 releaseATCAIface()	152
18.6 Certificate manipulation methods (atcacert_)	153
18.6.1 Detailed Description	158
18.6.2 Macro Definition Documentation	158
18.6.2.1 ATCA_PACKED	158
18.6.2.2 ATCACERT_DATE_FORMAT_SIZES_COUNT	158
18.6.2.3 ATCACERT_E_BAD_CERT	159
18.6.2.4 ATCACERT_E_BAD_PARAMS	159
18.6.2.5 ATCACERT_E_BUFFER_TOO_SMALL	159
18.6.2.6 ATCACERT_E_DECODING_ERROR	159
18.6.2.7 ATCACERT_E_ELEM_MISSING	159
18.6.2.8 ATCACERT_E_ELEM_OUT_OF_BOUNDS	159
18.6.2.9 ATCACERT_E_ERROR	160
18.6.2.10 ATCACERT_E_INVALID_DATE	160
18.6.2.11 ATCACERT_E_INVALID_TRANSFORM	160
18.6.2.12 ATCACERT_E_SUCCESS	160
18.6.2.13 ATCACERT_E_UNEXPECTED_ELEM_SIZE	160
18.6.2.14 ATCACERT_E_UNIMPLEMENTED	160
18.6.2.15 ATCACERT_E_VERIFY_FAILED	161
18.6.2.16 ATCACERT_E_WRONG_CERT_DEF	161
18.6.2.17 DATEFMT_ISO8601_SEP_SIZE	161
18.6.2.18 DATEFMT_MAX_SIZE	161
18.6.2.19 DATEFMT_POSIX_UINT32_BE_SIZE	161
18.6.2.20 DATEFMT_POSIX_UINT32_LE_SIZE	161
18.6.2.21 DATEFMT_RFC5280_GEN_SIZE	161
18.6.2.22 DATEFMT_RFC5280_UTC_SIZE	162
18.6.2.23 FALSE	162
18.6.2.24 TRUE	162
18.6.3 Typedef Documentation	162
18.6.3.1 atcacert_build_state_t	162
18.6.3.2 atcacert_cert_element_t	162
18.6.3.3 atcacert_cert_loc_t	162
18.6.3.4 atcacert_cert_sn_src_t	162
18.6.3.5 atcacert_cert_type_t	163
18.6.3.6 atcacert_date_format_t	163
18.6.3.7 atcacert_def_t	163
18.6.3.8 atcacert_device_loc_t	163
18.6.3.9 atcacert_device_zone_t	163
18.6.3.10 atcacert_std_cert_element_t	163

18.6.3.11 atcacert_tm_utc_t . . . . .	163
18.6.3.12 atcacert_transform_t . . . . .	164
18.6.4 Enumeration Type Documentation . . . . .	164
18.6.4.1 atcacert_cert_sn_src_e . . . . .	164
18.6.4.2 atcacert_cert_type_e . . . . .	165
18.6.4.3 atcacert_date_format_e . . . . .	165
18.6.4.4 atcacert_device_zone_e . . . . .	166
18.6.4.5 atcacert_std_cert_element_e . . . . .	166
18.6.4.6 atcacert_transform_e . . . . .	166
18.6.5 Function Documentation . . . . .	168
18.6.5.1 atcacert_cert_build_finish() . . . . .	168
18.6.5.2 atcacert_cert_build_process() . . . . .	168
18.6.5.3 atcacert_cert_build_start() . . . . .	169
18.6.5.4 atcacert_create_csr() . . . . .	169
18.6.5.5 atcacert_create_csr_pem() . . . . .	170
18.6.5.6 atcacert_date_dec() . . . . .	170
18.6.5.7 atcacert_date_dec_compcert() . . . . .	171
18.6.5.8 atcacert_date_dec_iso8601_sep() . . . . .	171
18.6.5.9 atcacert_date_dec_posix_uint32_be() . . . . .	172
18.6.5.10 atcacert_date_dec_posix_uint32_le() . . . . .	172
18.6.5.11 atcacert_date_dec_rfc5280_gen() . . . . .	172
18.6.5.12 atcacert_date_dec_rfc5280_utc() . . . . .	172
18.6.5.13 atcacert_date_enc() . . . . .	172
18.6.5.14 atcacert_date_enc_compcert() . . . . .	173
18.6.5.15 atcacert_date_enc_iso8601_sep() . . . . .	173
18.6.5.16 atcacert_date_enc_posix_uint32_be() . . . . .	173
18.6.5.17 atcacert_date_enc_posix_uint32_le() . . . . .	173
18.6.5.18 atcacert_date_enc_rfc5280_gen() . . . . .	174
18.6.5.19 atcacert_date_enc_rfc5280_utc() . . . . .	174
18.6.5.20 atcacert_date_get_max_date() . . . . .	174
18.6.5.21 atcacert_der_adjust_length() . . . . .	174
18.6.5.22 atcacert_der_dec_ecdsa_sig_value() . . . . .	175
18.6.5.23 atcacert_der_dec_integer() . . . . .	175
18.6.5.24 atcacert_der_dec_length() . . . . .	176
18.6.5.25 atcacert_der_enc_ecdsa_sig_value() . . . . .	176
18.6.5.26 atcacert_der_enc_integer() . . . . .	177
18.6.5.27 atcacert_der_enc_length() . . . . .	177
18.6.5.28 atcacert_gen_cert_sn() . . . . .	178
18.6.5.29 atcacert_gen_challenge_hw() . . . . .	178
18.6.5.30 atcacert_gen_challenge_sw() . . . . .	179
18.6.5.31 atcacert_get_auth_key_id() . . . . .	179
18.6.5.32 atcacert_get_cert_element() . . . . .	179

18.6.5.33 atcacert_get_cert_sn()	180
18.6.5.34 atcacert_get_comp_cert()	181
18.6.5.35 atcacert_get_device_data()	181
18.6.5.36 atcacert_get_device_locs()	182
18.6.5.37 atcacert_get_expire_date()	182
18.6.5.38 atcacert_get_issue_date()	183
18.6.5.39 atcacert_get_key_id()	183
18.6.5.40 atcacert_get_response()	184
18.6.5.41 atcacert_get_signature()	184
18.6.5.42 atcacert_get_signer_id()	185
18.6.5.43 atcacert_get_subj_key_id()	185
18.6.5.44 atcacert_get_subj_public_key()	186
18.6.5.45 atcacert_get_tbs()	186
18.6.5.46 atcacert_get_tbs_digest()	187
18.6.5.47 atcacert_is_device_loc_overlap()	187
18.6.5.48 atcacert_max_cert_size()	188
18.6.5.49 atcacert_merge_device_loc()	188
18.6.5.50 atcacert_public_key_add_padding()	189
18.6.5.51 atcacert_public_key_remove_padding()	189
18.6.5.52 atcacert_read_cert()	189
18.6.5.53 atcacert_read_cert_size()	191
18.6.5.54 atcacert_read_device_loc()	191
18.6.5.55 atcacert_read_subj_key_id()	192
18.6.5.56 atcacert_set_auth_key_id()	192
18.6.5.57 atcacert_set_auth_key_id_raw()	193
18.6.5.58 atcacert_set_cert_element()	193
18.6.5.59 atcacert_set_cert_sn()	194
18.6.5.60 atcacert_set_comp_cert()	194
18.6.5.61 atcacert_set_expire_date()	195
18.6.5.62 atcacert_set_issue_date()	195
18.6.5.63 atcacert_set_signature()	196
18.6.5.64 atcacert_set_signer_id()	196
18.6.5.65 atcacert_set_subj_public_key()	197
18.6.5.66 atcacert_transform_data()	197
18.6.5.67 atcacert_verify_cert_hw()	198
18.6.5.68 atcacert_verify_cert_sw()	198
18.6.5.69 atcacert_verify_response_hw()	199
18.6.5.70 atcacert_verify_response_sw()	199
18.6.5.71 atcacert_write_cert()	200
18.6.6 Variable Documentation	200
18.6.6.1 ATCACERT_DATE_FORMAT_SIZES	200
18.7 Basic Crypto API methods for CryptoAuth Devices (calib_)	201

18.7.1 Detailed Description . . . . .	206
18.7.2 Typedef Documentation . . . . .	206
18.7.2.1 atca_hmac_sha256_ctx_t . . . . .	206
18.7.2.2 atca_sha256_ctx_t . . . . .	206
18.7.3 Function Documentation . . . . .	206
18.7.3.1 _calib_exit() . . . . .	206
18.7.3.2 calib_aes() . . . . .	207
18.7.3.3 calib_aes_decrypt() . . . . .	207
18.7.3.4 calib_aes_encrypt() . . . . .	208
18.7.3.5 calib_aes_gfm() . . . . .	208
18.7.3.6 calib_cfg_discover() . . . . .	209
18.7.3.7 calib_challenge() . . . . .	209
18.7.3.8 calib_challenge_seed_update() . . . . .	210
18.7.3.9 calib_checkmac() . . . . .	210
18.7.3.10 calib_cmp_config_zone() . . . . .	211
18.7.3.11 calib_counter() . . . . .	211
18.7.3.12 calib_counter_increment() . . . . .	212
18.7.3.13 calib_counter_read() . . . . .	212
18.7.3.14 calib_derivekey() . . . . .	212
18.7.3.15 calib_ecdh() . . . . .	213
18.7.3.16 calib_ecdh_base() . . . . .	213
18.7.3.17 calib_ecdh_enc() . . . . .	214
18.7.3.18 calib_ecdh_ioenc() . . . . .	214
18.7.3.19 calib_ecdh_tempkey() . . . . .	215
18.7.3.20 calib_ecdh_tempkey_ioenc() . . . . .	215
18.7.3.21 calib_gendig() . . . . .	216
18.7.3.22 calib_genkey() . . . . .	216
18.7.3.23 calib_genkey_base() . . . . .	217
18.7.3.24 calib_get_addr() . . . . .	217
18.7.3.25 calib_get_pubkey() . . . . .	218
18.7.3.26 calib_get_zone_size() . . . . .	218
18.7.3.27 calib_hmac() . . . . .	218
18.7.3.28 calib_hw_sha2_256() . . . . .	219
18.7.3.29 calib_hw_sha2_256_finish() . . . . .	219
18.7.3.30 calib_hw_sha2_256_init() . . . . .	220
18.7.3.31 calib_hw_sha2_256_update() . . . . .	220
18.7.3.32 calib_idle() . . . . .	221
18.7.3.33 calib_info() . . . . .	221
18.7.3.34 calib_info_base() . . . . .	221
18.7.3.35 calib_info_get_latch() . . . . .	222
18.7.3.36 calib_info_set_latch() . . . . .	222
18.7.3.37 calib_is_locked() . . . . .	223

18.7.3.38 calib_is_slot_locked()	223
18.7.3.39 calib_kdf()	224
18.7.3.40 calib_lock()	224
18.7.3.41 calib_lock_config_zone()	225
18.7.3.42 calib_lock_config_zone_crc()	225
18.7.3.43 calib_lock_data_slot()	225
18.7.3.44 calib_lock_data_zone()	226
18.7.3.45 calib_lock_data_zone_crc()	226
18.7.3.46 calib_mac()	227
18.7.3.47 calib_nonce()	227
18.7.3.48 calib_nonce_base()	228
18.7.3.49 calib_nonce_load()	228
18.7.3.50 calib_nonce_rand()	229
18.7.3.51 calib_priv_write()	229
18.7.3.52 calib_random()	229
18.7.3.53 calib_read_bytes_zone()	230
18.7.3.54 calib_read_config_zone()	230
18.7.3.55 calib_read_enc()	231
18.7.3.56 calib_read_pubkey()	231
18.7.3.57 calib_read_serial_number()	232
18.7.3.58 calib_read_sig()	232
18.7.3.59 calib_read_zone()	232
18.7.3.60 calib_secureboot()	233
18.7.3.61 calib_secureboot_mac()	234
18.7.3.62 calib_selftest()	234
18.7.3.63 calib_sha()	235
18.7.3.64 calib_sha_base()	235
18.7.3.65 calib_sha_end()	236
18.7.3.66 calib_sha_hmac()	236
18.7.3.67 calib_sha_hmac_finish()	237
18.7.3.68 calib_sha_hmac_init()	237
18.7.3.69 calib_sha_hmac_update()	238
18.7.3.70 calib_sha_read_context()	238
18.7.3.71 calib_sha_start()	239
18.7.3.72 calib_sha_update()	239
18.7.3.73 calib_sha_write_context()	239
18.7.3.74 calib_sign()	240
18.7.3.75 calib_sign_base()	240
18.7.3.76 calib_sign_internal()	241
18.7.3.77 calib_sleep()	241
18.7.3.78 calib_updateextra()	242
18.7.3.79 calib_verify()	242

18.7.3.80 calib_verify_extern()	243
18.7.3.81 calib_verify_extern_mac()	244
18.7.3.82 calib_verify_invalidate()	244
18.7.3.83 calib_verify_stored()	245
18.7.3.84 calib_verify_stored_mac()	245
18.7.3.85 calib_verify_validate()	246
18.7.3.86 calib_wakeup()	246
18.7.3.87 calib_write()	247
18.7.3.88 calib_write_bytes_zone()	247
18.7.3.89 calib_write_config_counter()	248
18.7.3.90 calib_write_config_zone()	248
18.7.3.91 calib_write_enc()	249
18.7.3.92 calib_write_pubkey()	249
18.7.3.93 calib_write_zone()	249
18.7.4 Variable Documentation	250
18.7.4.1 atca_basic_aes_gcm_version	250
18.8 Software crypto methods (atcac_)	251
18.8.1 Detailed Description	251
18.8.2 Macro Definition Documentation	251
18.8.2.1 ATCA_ECC_P256_FIELD_SIZE	252
18.8.2.2 ATCA_ECC_P256_PRIVATE_KEY_SIZE	252
18.8.2.3 ATCA_ECC_P256_PUBLIC_KEY_SIZE	252
18.8.2.4 ATCA_ECC_P256_SIGNATURE_SIZE	252
18.8.3 Function Documentation	252
18.8.3.1 atcac_sha256_hmac_finish()	252
18.8.3.2 atcac_sha256_hmac_init()	253
18.8.3.3 atcac_sha256_hmac_update()	253
18.8.3.4 atcac_sw_ecdsa_verify_p256()	253
18.8.3.5 atcac_sw_random()	254
18.8.3.6 atcac_sw_sha1()	254
18.8.3.7 atcac_sw_sha1_finish()	254
18.8.3.8 atcac_sw_sha1_init()	254
18.8.3.9 atcac_sw_sha1_update()	255
18.8.3.10 atcac_sw_sha2_256()	255
18.8.3.11 atcac_sw_sha2_256_finish()	255
18.8.3.12 atcac_sw_sha2_256_init()	256
18.8.3.13 atcac_sw_sha2_256_update()	256
18.9 Hardware abstraction layer (hal_)	257
18.9.1 Detailed Description	262
18.9.2 Macro Definition Documentation	262
18.9.2.1 ATCA_POLLING_FREQUENCY_TIME_MSEC	262
18.9.2.2 ATCA_POLLING_INIT_TIME_MSEC	262

18.9.2.3 ATCA_POLLING_MAX_TIME_MSEC . . . . .	262
18.9.2.4 DEBUG_PIN_1 . . . . .	263
18.9.2.5 DEBUG_PIN_2 . . . . .	263
18.9.2.6 hal_memset_s . . . . .	263
18.9.2.7 HID_DEVICES_MAX [1/3] . . . . .	263
18.9.2.8 HID_DEVICES_MAX [2/3] . . . . .	263
18.9.2.9 HID_DEVICES_MAX [3/3] . . . . .	263
18.9.2.10 HID_GUID . . . . .	263
18.9.2.11 HID_PACKET_MAX [1/3] . . . . .	264
18.9.2.12 HID_PACKET_MAX [2/3] . . . . .	264
18.9.2.13 HID_PACKET_MAX [3/3] . . . . .	264
18.9.2.14 KIT_MAX_SCAN_COUNT . . . . .	264
18.9.2.15 KIT_MAX_TX_BUF . . . . .	264
18.9.2.16 KIT_MSG_SIZE . . . . .	264
18.9.2.17 KIT_RX_WRAP_SIZE . . . . .	264
18.9.2.18 KIT_TX_WRAP_SIZE . . . . .	264
18.9.2.19 MAX_I2C_BUSES [1/2] . . . . .	265
18.9.2.20 MAX_I2C_BUSES [2/2] . . . . .	265
18.9.2.21 MAX_SPI_BUSES . . . . .	265
18.9.2.22 MAX_SWI_BUSES [1/2] . . . . .	265
18.9.2.23 MAX_SWI_BUSES [2/2] . . . . .	265
18.9.2.24 RECEIVE_MODE [1/2] . . . . .	265
18.9.2.25 RECEIVE_MODE [2/2] . . . . .	266
18.9.2.26 RX_DELAY [1/2] . . . . .	266
18.9.2.27 RX_DELAY [2/2] . . . . .	266
18.9.2.28 SWI_FLAG_CMD . . . . .	266
18.9.2.29 SWI_FLAG_IDLE . . . . .	266
18.9.2.30 SWI_FLAG_SLEEP . . . . .	266
18.9.2.31 SWI_FLAG_TX . . . . .	266
18.9.2.32 SWI_WAKE_TOKEN . . . . .	267
18.9.2.33 TRANSMIT_MODE [1/2] . . . . .	267
18.9.2.34 TRANSMIT_MODE [2/2] . . . . .	267
18.9.2.35 TX_DELAY [1/2] . . . . .	267
18.9.2.36 TX_DELAY [2/2] . . . . .	267
18.9.3 Typedef Documentation . . . . .	267
18.9.3.1 atcahid_t [1/3] . . . . .	267
18.9.3.2 atcahid_t [2/3] . . . . .	267
18.9.3.3 atcahid_t [3/3] . . . . .	268
18.9.3.4 ATCAI2CMaster_t [1/2] . . . . .	268
18.9.3.5 ATCAI2CMaster_t [2/2] . . . . .	268
18.9.3.6 ATCASPIMaster_t . . . . .	268
18.9.3.7 ATCASWIMaster_t [1/2] . . . . .	268



18.9.3.8 ATCASWIMaster_t [2/2]	268
18.9.3.9 hid_device_t [1/2]	268
18.9.3.10 hid_device_t [2/2]	269
18.9.3.11 i2c_sam_instance_t	269
18.9.3.12 i2c_start_instance_t	269
18.9.3.13 sam_change_baudrate	269
18.9.3.14 start_change_baudrate	269
18.9.4 Function Documentation	269
18.9.4.1 atca_delay_10us()	269
18.9.4.2 atca_delay_ms()	270
18.9.4.3 atca_delay_us()	270
18.9.4.4 change_i2c_speed()	270
18.9.4.5 hal_check_wake()	271
18.9.4.6 hal_create_mutex()	271
18.9.4.7 hal_delay_10us()	271
18.9.4.8 hal_delay_ms()	272
18.9.4.9 hal_delay_us()	272
18.9.4.10 hal_destroy_mutex()	272
18.9.4.11 hal_free()	272
18.9.4.12 hal_i2c_discover_buses()	273
18.9.4.13 hal_i2c_discover_devices()	274
18.9.4.14 hal_i2c_idle()	275
18.9.4.15 hal_i2c_init()	275
18.9.4.16 hal_i2c_post_init()	277
18.9.4.17 hal_i2c_receive()	277
18.9.4.18 hal_i2c_release()	278
18.9.4.19 hal_i2c_send()	279
18.9.4.20 hal_i2c_sleep()	280
18.9.4.21 hal_i2c_wake()	280
18.9.4.22 hal_iface_init()	281
18.9.4.23 hal_iface_register_hal()	281
18.9.4.24 hal_iface_release()	281
18.9.4.25 hal_kit_hid_discover_buses()	282
18.9.4.26 hal_kit_hid_discover_devices()	283
18.9.4.27 hal_kit_hid_idle()	283
18.9.4.28 hal_kit_hid_init()	284
18.9.4.29 hal_kit_hid_post_init()	284
18.9.4.30 hal_kit_hid_receive()	285
18.9.4.31 hal_kit_hid_release()	286
18.9.4.32 hal_kit_hid_send()	286
18.9.4.33 hal_kit_hid_sleep()	287
18.9.4.34 hal_kit_hid_wake()	287

18.9.4.35 hal_lock_mutex()	288
18.9.4.36 hal_malloc()	288
18.9.4.37 hal_rtos_delay_ms()	288
18.9.4.38 hal_spi_discover_buses()	289
18.9.4.39 hal_spi_discover_devices()	289
18.9.4.40 hal_spi_idle()	289
18.9.4.41 hal_spi_init()	290
18.9.4.42 hal_spi_post_init()	290
18.9.4.43 hal_spi_receive()	291
18.9.4.44 hal_spi_release()	291
18.9.4.45 hal_spi_send()	291
18.9.4.46 hal_spi_sleep()	292
18.9.4.47 hal_spi_wake()	292
18.9.4.48 hal_swi_discover_buses()	293
18.9.4.49 hal_swi_discover_devices()	293
18.9.4.50 hal_swi_idle()	293
18.9.4.51 hal_swi_init()	294
18.9.4.52 hal_swi_post_init()	294
18.9.4.53 hal_swi_receive()	295
18.9.4.54 hal_swi_release()	295
18.9.4.55 hal_swi_send()	295
18.9.4.56 hal_swi_send_flag()	296
18.9.4.57 hal_swi_sleep()	296
18.9.4.58 hal_swi_wake()	297
18.9.4.59 hal_unlock_mutex()	297
18.9.4.60 kit_id_from_devtype()	297
18.9.4.61 kit_idle()	297
18.9.4.62 kit_init()	298
18.9.4.63 kit_interface_from_kittype()	298
18.9.4.64 kit_parse_rsp()	298
18.9.4.65 kit_phy_num_found()	299
18.9.4.66 kit_phy_receive() [1/2]	299
18.9.4.67 kit_phy_receive() [2/2]	300
18.9.4.68 kit_phy_send() [1/2]	300
18.9.4.69 kit_phy_send() [2/2]	301
18.9.4.70 kit_receive()	301
18.9.4.71 kit_send()	302
18.9.4.72 kit_sleep()	302
18.9.4.73 kit_wake()	303
18.9.4.74 kit_wrap_cmd()	303
18.9.4.75 strnchr()	304
18.9.4.76 swi_uart_deinit()	304

18.9.4.77 swi_uart_discover_buses()	304
18.9.4.78 swi_uart_init()	305
18.9.4.79 swi_uart_mode()	305
18.9.4.80 swi_uart_receive_byte()	306
18.9.4.81 swi_uart_send_byte()	306
18.9.4.82 swi_uart_setbaud()	306
18.9.5 Variable Documentation	307
18.9.5.1 _gHid [1/3]	307
18.9.5.2 _gHid [2/3]	307
18.9.5.3 _gHid [3/3]	307
18.9.5.4 pin_conf	307
18.10 Host side crypto methods (atcah_)	308
18.10.1 Detailed Description	312
18.10.2 Macro Definition Documentation	312
18.10.2.1 ATCA_COMMAND_HEADER_SIZE	312
18.10.2.2 ATCA_DERIVE_KEY_ZEROS_SIZE	312
18.10.2.3 ATCA_GENDIG_ZEROS_SIZE	313
18.10.2.4 ATCA_HMAC_BLOCK_SIZE	313
18.10.2.5 ATCA_MSG_SIZE_DERIVE_KEY	313
18.10.2.6 ATCA_MSG_SIZE_DERIVE_KEY_MAC	313
18.10.2.7 ATCA_MSG_SIZE_ENCRYPT_MAC	313
18.10.2.8 ATCA_MSG_SIZE_GEN_DIG	313
18.10.2.9 ATCA_MSG_SIZE_HMAC	313
18.10.2.10 ATCA_MSG_SIZE_MAC	314
18.10.2.11 ATCA_MSG_SIZE_NONCE	314
18.10.2.12 ATCA_MSG_SIZE_PRIVWRITE_MAC	314
18.10.2.13 ATCA_PRIVWRITE_MAC_ZEROS_SIZE	314
18.10.2.14 ATCA_PRIVWRITE_PLAIN_TEXT_SIZE	314
18.10.2.15 ATCA_SN_0_DEF	314
18.10.2.16 ATCA_SN_1_DEF	314
18.10.2.17 ATCA_SN_8_DEF	315
18.10.2.18 ATCA_WRITE_MAC_ZEROS_SIZE	315
18.10.2.19 ENCRYPTION_KEY_SIZE	315
18.10.2.20 MAC_MODE_USE_TEMPKEY_MASK	315
18.10.3 Typedef Documentation	315
18.10.3.1 atca_check_mac_in_out_t	315
18.10.3.2 atca_gen_dig_in_out_t	315
18.10.3.3 atca_gen_key_in_out_t	316
18.10.3.4 atca_io_decrypt_in_out_t	316
18.10.3.5 atca_mac_in_out_t	316
18.10.3.6 atca_nonce_in_out_t	316
18.10.3.7 atca_secureboot_enc_in_out_t	316

18.10.3.8 atca_secureboot_mac_in_out_t . . . . .	316
18.10.3.9 atca_sign_internal_in_out_t . . . . .	316
18.10.3.10 atca_temp_key_t . . . . .	317
18.10.3.11 atca_verify_in_out_t . . . . .	317
18.10.3.12 atca_verify_mac_in_out_t . . . . .	317
18.10.3.13 atca_write_mac_in_out_t . . . . .	317
18.10.4 Function Documentation . . . . .	317
18.10.4.1 atcah_check_mac() . . . . .	317
18.10.4.2 atcah_config_to_sign_internal() . . . . .	318
18.10.4.3 atcah_decrypt() . . . . .	318
18.10.4.4 atcah_derive_key() . . . . .	319
18.10.4.5 atcah_derive_key_mac() . . . . .	319
18.10.4.6 atcah_encode_counter_match() . . . . .	320
18.10.4.7 atcah_gen_dig() . . . . .	320
18.10.4.8 atcah_gen_key_msg() . . . . .	320
18.10.4.9 atcah_gen_mac() . . . . .	321
18.10.4.10 atcah_hmac() . . . . .	321
18.10.4.11 atcah_include_data() . . . . .	322
18.10.4.12 atcah_io_decrypt() . . . . .	322
18.10.4.13 atcah_mac() . . . . .	322
18.10.4.14 atcah_nonce() . . . . .	323
18.10.4.15 atcah_privwrite_auth_mac() . . . . .	323
18.10.4.16 atcah_secureboot_enc() . . . . .	323
18.10.4.17 atcah_secureboot_mac() . . . . .	324
18.10.4.18 atcah_sha256() . . . . .	324
18.10.4.19 atcah_sign_internal_msg() . . . . .	325
18.10.4.20 atcah_verify_mac() . . . . .	325
18.10.4.21 atcah_write_auth_mac() . . . . .	325
18.10.5 Variable Documentation . . . . .	326
18.10.5.1 challenge . . . . .	326
18.10.5.2 crypto_data . . . . .	326
18.10.5.3 curve_type . . . . .	326
18.10.5.4 key [1/2] . . . . .	326
18.10.5.5 key [2/2] . . . . .	327
18.10.5.6 key_id [1/2] . . . . .	327
18.10.5.7 key_id [2/2] . . . . .	327
18.10.5.8 mode [1/3] . . . . .	327
18.10.5.9 mode [2/3] . . . . .	327
18.10.5.10 mode [3/3] . . . . .	327
18.10.5.11 num_in . . . . .	328
18.10.5.12 otp [1/3] . . . . .	328
18.10.5.13 otp [2/3] . . . . .	328

18.10.5.14 otp [3/3]	328
18.10.5.15 p_temp	328
18.10.5.16 public_key	328
18.10.5.17 rand_out	329
18.10.5.18 response [1/2]	329
18.10.5.19 response [2/2]	329
18.10.5.20 signature	329
18.10.5.21 sn [1/3]	329
18.10.5.22 sn [2/3]	329
18.10.5.23 sn [3/3]	330
18.10.5.24 temp_key [1/5]	330
18.10.5.25 temp_key [2/5]	330
18.10.5.26 temp_key [3/5]	330
18.10.5.27 temp_key [4/5]	330
18.10.5.28 temp_key [5/5]	330
18.10.5.29 zero	330
18.11 JSON Web Token (JWT) methods (atca_jwt_)	331
18.11.1 Detailed Description	331
18.11.2 Function Documentation	331
18.11.2.1 atca_jwt_add_claim_numeric()	331
18.11.2.2 atca_jwt_add_claim_string()	332
18.11.2.3 atca_jwt_check_payload_start()	332
18.11.2.4 atca_jwt_finalize()	332
18.11.2.5 atca_jwt_init()	332
18.11.2.6 atca_jwt_verify()	332
18.12 mbedTLS Wrapper methods (atca_mbedtls_)	333
18.12.1 Detailed Description	333
18.12.2 Function Documentation	333
18.12.2.1 atca_mbedtls_cert_add()	333
18.12.2.2 atca_mbedtls_ecdh_ioprot_cb()	333
18.12.2.3 atca_mbedtls_ecdh_slot_cb()	334
18.12.2.4 atca_mbedtls_pk_init()	334
18.13 Attributes (pkcs11_attr_)	335
18.13.1 Detailed Description	342
18.13.2 Macro Definition Documentation	342
18.13.2.1 PKCS11_MECH_ECC508_EC_CAPABILITY	342
18.13.2.2 TABLE_SIZE	342
18.13.3 Typedef Documentation	342
18.13.3.1 pkcs11_mech_table_e	342
18.13.3.2 pkcs11_mech_table_ptr	343
18.13.4 Function Documentation	343
18.13.4.1 C_CancelFunction()	343

18.13.4.2 C_CloseAllSessions()	343
18.13.4.3 C_CloseSession()	343
18.13.4.4 C_CopyObject()	343
18.13.4.5 C_CreateObject()	344
18.13.4.6 C_Decrypt()	344
18.13.4.7 C_DecryptDigestUpdate()	344
18.13.4.8 C_DecryptFinal()	344
18.13.4.9 C_DecryptInit()	345
18.13.4.10 C_DecryptUpdate()	345
18.13.4.11 C_DecryptVerifyUpdate()	345
18.13.4.12 C_DeriveKey()	345
18.13.4.13 C_DestroyObject()	346
18.13.4.14 C_Digest()	346
18.13.4.15 C_DigestEncryptUpdate()	346
18.13.4.16 C_DigestFinal()	346
18.13.4.17 C_DigestInit()	347
18.13.4.18 C_DigestKey()	347
18.13.4.19 C_DigestUpdate()	347
18.13.4.20 C_Encrypt()	347
18.13.4.21 C_EncryptFinal()	347
18.13.4.22 C_EncryptInit()	348
18.13.4.23 C_EncryptUpdate()	348
18.13.4.24 C_Finalize()	348
18.13.4.25 C_FindObjects()	348
18.13.4.26 C_FindObjectsFinal()	348
18.13.4.27 C_FindObjectsInit()	349
18.13.4.28 C_GenerateKey()	349
18.13.4.29 C_GenerateKeyPair()	349
18.13.4.30 C_GenerateRandom()	349
18.13.4.31 C_GetAttributeValue()	350
18.13.4.32 C_GetFunctionList()	350
18.13.4.33 C_GetFunctionStatus()	350
18.13.4.34 C_GetInfo()	350
18.13.4.35 C_GetMechanismInfo()	350
18.13.4.36 C_GetMechanismList()	351
18.13.4.37 C_GetObjectSize()	351
18.13.4.38 C_GetOperationState()	351
18.13.4.39 C_GetSessionInfo()	351
18.13.4.40 C_GetSlotInfo()	351
18.13.4.41 C_GetSlotList()	352
18.13.4.42 C_GetTokenInfo()	352
18.13.4.43 C_Initialize()	352

18.13.4.44 C_InitPIN()	352
18.13.4.45 C_InitToken()	352
18.13.4.46 C_Login()	353
18.13.4.47 C_Logout()	353
18.13.4.48 C_OpenSession()	353
18.13.4.49 C_SeedRandom()	353
18.13.4.50 C_SetAttributeValue()	354
18.13.4.51 C_SetOperationState()	354
18.13.4.52 C_SetPIN()	354
18.13.4.53 C_Sign()	354
18.13.4.54 C_SignEncryptUpdate()	355
18.13.4.55 C_SignFinal()	355
18.13.4.56 C_SignInit()	355
18.13.4.57 C_SignRecover()	355
18.13.4.58 C_SignRecoverInit()	356
18.13.4.59 C_SignUpdate()	356
18.13.4.60 C_UnwrapKey()	356
18.13.4.61 C_Verify()	356
18.13.4.62 C_VerifyFinal()	357
18.13.4.63 C_VerifyInit()	357
18.13.4.64 C_VerifyRecover()	357
18.13.4.65 C_VerifyRecoverInit()	357
18.13.4.66 C_VerifyUpdate()	357
18.13.4.67 C_WaitForSlotEvent()	358
18.13.4.68 C_WrapKey()	358
18.13.4.69 pkcs11_attrib_empty()	358
18.13.4.70 pkcs11_attrib_false()	358
18.13.4.71 pkcs11_attrib_fill()	358
18.13.4.72 pkcs11_attrib_true()	359
18.13.4.73 pkcs11_attrib_value()	359
18.13.4.74 pkcs11_cert_get_authority_key_id()	359
18.13.4.75 pkcs11_cert_get_encoded()	359
18.13.4.76 pkcs11_cert_get_subject()	359
18.13.4.77 pkcs11_cert_get_subject_key_id()	359
18.13.4.78 pkcs11_cert_get_trusted_flag()	360
18.13.4.79 pkcs11_cert_get_type()	360
18.13.4.80 pkcs11_cert_x509_write()	360
18.13.4.81 pkcs11_config_cert()	360
18.13.4.82 pkcs11_config_init_cert()	360
18.13.4.83 pkcs11_config_init_private()	360
18.13.4.84 pkcs11_config_init_public()	361
18.13.4.85 pkcs11_config_key()	361

18.13.4.86 pkcs11_config_load()	361
18.13.4.87 pkcs11_config_load_objects()	361
18.13.4.88 pkcs11_config_remove_object()	361
18.13.4.89 pkcs11_deinit()	361
18.13.4.90 pkcs11_find_continue()	362
18.13.4.91 pkcs11_find_finish()	362
18.13.4.92 pkcs11_find_get_attribute()	362
18.13.4.93 pkcs11_find_init()	362
18.13.4.94 pkcs11_get_context()	362
18.13.4.95 pkcs11_get_lib_info()	362
18.13.4.96 pkcs11_get_session_context()	363
18.13.4.97 pkcs11_init()	363
18.13.4.98 pkcs11_init_check()	363
18.13.4.99 pkcs11_key_derive()	363
18.13.4.100 pkcs11_key_generate_pair()	364
18.13.4.101 pkcs11_key_write()	364
18.13.4.102 pkcs11_lock_context()	364
18.13.4.103 pkcs11_mech_get_list()	364
18.13.4.104 pkcs11_object_alloc()	364
18.13.4.105 pkcs11_object_check()	365
18.13.4.106 pkcs11_object_create()	365
18.13.4.107 pkcs11_object_deinit()	365
18.13.4.108 pkcs11_object_destroy()	365
18.13.4.109 pkcs11_object_find()	365
18.13.4.110 pkcs11_object_free()	365
18.13.4.111 pkcs11_object_get_class()	366
18.13.4.112 pkcs11_object_get_destroyable()	366
18.13.4.113 pkcs11_object_get_handle()	366
18.13.4.114 pkcs11_object_get_name()	366
18.13.4.115 pkcs11_object_get_size()	366
18.13.4.116 pkcs11_object_get_type()	366
18.13.4.117 pkcs11_object_load_handle_info()	367
18.13.4.118 pkcs11_os_create_mutex()	367
18.13.4.119 pkcs11_os_destroy_mutex()	367
18.13.4.120 pkcs11_os_lock_mutex()	367
18.13.4.121 pkcs11_os_unlock_mutex()	367
18.13.4.122 pkcs11_session_check()	367
18.13.4.123 pkcs11_session_close()	368
18.13.4.124 pkcs11_session_closeall()	368
18.13.4.125 pkcs11_session_get_info()	368
18.13.4.126 pkcs11_session_login()	368
18.13.4.127 pkcs11_session_logout()	368



18.13.4.128	<a href="#">pkcs11_session_open()</a>	368
18.13.4.129	<a href="#">pkcs11_signature_sign()</a>	369
18.13.4.130	<a href="#">pkcs11_signature_sign_continue()</a>	369
18.13.4.131	<a href="#">pkcs11_signature_sign_finish()</a>	369
18.13.4.132	<a href="#">pkcs11_signature_sign_init()</a>	369
18.13.4.133	<a href="#">pkcs11_signature_verify()</a>	370
18.13.4.134	<a href="#">pkcs11_signature_verify_continue()</a>	370
18.13.4.135	<a href="#">pkcs11_signature_verify_finish()</a>	370
18.13.4.136	<a href="#">pkcs11_signature_verify_init()</a>	370
18.13.4.137	<a href="#">pkcs11_slot_config()</a>	370
18.13.4.138	<a href="#">pkcs11_slot_get_context()</a>	371
18.13.4.139	<a href="#">pkcs11_slot_get_info()</a>	371
18.13.4.140	<a href="#">pkcs11_slot_get_list()</a>	371
18.13.4.141	<a href="#">pkcs11_slot_init()</a>	371
18.13.4.142	<a href="#">pkcs11_slot_initslots()</a>	371
18.13.4.143	<a href="#">pkcs11_token_convert_pin_to_key()</a>	371
18.13.4.144	<a href="#">pkcs11_token_get_access_type()</a>	372
18.13.4.145	<a href="#">pkcs11_token_get_info()</a>	372
18.13.4.146	<a href="#">pkcs11_token_get_storage()</a>	372
18.13.4.147	<a href="#">pkcs11_token_get_writable()</a>	372
18.13.4.148	<a href="#">pkcs11_token_init()</a>	372
18.13.4.149	<a href="#">pkcs11_token_random()</a>	372
18.13.4.150	<a href="#">pkcs11_token_set_pin()</a>	373
18.13.4.151	<a href="#">pkcs11_unlock_context()</a>	373
18.13.4.152	<a href="#">pkcs11_util_convert_rv()</a>	373
18.13.4.153	<a href="#">pkcs11_util_escape_string()</a>	373
18.13.4.154	<a href="#">pkcs11_util_memset()</a>	373
18.13.4.155	<a href="#">pkcs_mech_get_info()</a>	373
18.13.5	<a href="#">Variable Documentation</a>	374
18.13.5.1	<a href="#">pkcs11_cert_wtlspublic_attributes</a>	374
18.13.5.2	<a href="#">pkcs11_cert_wtlspublic_attributes_count</a>	374
18.13.5.3	<a href="#">pkcs11_cert_x509_attributes</a>	374
18.13.5.4	<a href="#">pkcs11_cert_x509_attributes_count</a>	374
18.13.5.5	<a href="#">pkcs11_cert_x509public_attributes</a>	374
18.13.5.6	<a href="#">pkcs11_cert_x509public_attributes_count</a>	374
18.13.5.7	<a href="#">pkcs11_key_ec_private_attributes</a>	375
18.13.5.8	<a href="#">pkcs11_key_ec_public_attributes</a>	375
18.13.5.9	<a href="#">pkcs11_key_private_attributes</a>	375
18.13.5.10	<a href="#">pkcs11_key_private_attributes_count</a>	375
18.13.5.11	<a href="#">pkcs11_key_public_attributes</a>	375
18.13.5.12	<a href="#">pkcs11_key_public_attributes_count</a>	375
18.13.5.13	<a href="#">pkcs11_key_rsa_private_attributes</a>	376

18.13.5.14 pkcs11_key_secret_attributes . . . . .	376
18.13.5.15 pkcs11_key_secret_attributes_count . . . . .	376
18.13.5.16 pkcs11_lib_description . . . . .	376
18.13.5.17 pkcs11_lib_manufacturer_id . . . . .	376
18.13.5.18 pkcs11_object_cache . . . . .	376
18.13.5.19 pkcs11_object_monotonic_attributes . . . . .	377
18.13.5.20 pkcs11_object_monotonic_attributes_count . . . . .	377
18.14 TNG API (tng_) . . . . .	378
18.14.1 Detailed Description . . . . .	379
18.14.2 Macro Definition Documentation . . . . .	379
18.14.2.1 CRYPTOAUTH_ROOT_CA_002_PUBLIC_KEY_OFFSET . . . . .	379
18.14.2.2 TNGLORA_CERT_TEMPLATE_4_DEVICE_SIZE . . . . .	379
18.14.2.3 TNGTLS_CERT_ELEMENTS_2_DEVICE_COUNT . . . . .	379
18.14.2.4 TNGTLS_CERT_TEMPLATE_1_SIGNER_SIZE . . . . .	379
18.14.2.5 TNGTLS_CERT_TEMPLATE_2_DEVICE_SIZE . . . . .	379
18.14.2.6 TNGTLS_CERT_TEMPLATE_3_DEVICE_SIZE . . . . .	379
18.14.3 Function Documentation . . . . .	379
18.14.3.1 tng_atcacert_device_public_key() . . . . .	379
18.14.3.2 tng_atcacert_max_device_cert_size() . . . . .	380
18.14.3.3 tng_atcacert_max_signer_cert_size() . . . . .	380
18.14.3.4 tng_atcacert_read_device_cert() . . . . .	381
18.14.3.5 tng_atcacert_read_signer_cert() . . . . .	381
18.14.3.6 tng_atcacert_root_cert() . . . . .	381
18.14.3.7 tng_atcacert_root_cert_size() . . . . .	382
18.14.3.8 tng_atcacert_root_public_key() . . . . .	382
18.14.3.9 tng_atcacert_signer_public_key() . . . . .	383
18.14.3.10 tng_get_device_cert_def() . . . . .	383
18.14.3.11 tng_get_device_pubkey() . . . . .	383
18.14.3.12 tng_map_get_device_cert_def() . . . . .	384
18.14.4 Variable Documentation . . . . .	384
18.14.4.1 g_cryptoauth_root_ca_002_cert . . . . .	384
18.14.4.2 g_cryptoauth_root_ca_002_cert_size . . . . .	384
18.14.4.3 g_tflxtls_cert_def_4_device . . . . .	384
18.14.4.4 g_tnglora_cert_def_1_signer . . . . .	384
18.14.4.5 g_tnglora_cert_def_2_device . . . . .	385
18.14.4.6 g_tnglora_cert_def_4_device . . . . .	385
18.14.4.7 g_tngtls_cert_def_1_signer . . . . .	385
18.14.4.8 g_tngtls_cert_def_2_device . . . . .	385
18.14.4.9 g_tngtls_cert_def_3_device . . . . .	385
<b>19 Data Structure Documentation</b>	<b>387</b>
19.1 _atecc508a_config Struct Reference . . . . .	387

19.1.1 Field Documentation . . . . .	387
19.1.1.1 ChipMode . . . . .	388
19.1.1.2 Counter0 . . . . .	388
19.1.1.3 Counter1 . . . . .	388
19.1.1.4 I2C_Address . . . . .	388
19.1.1.5 I2C_Enable . . . . .	388
19.1.1.6 KeyConfig . . . . .	388
19.1.1.7 LastKeyUse . . . . .	388
19.1.1.8 LockConfig . . . . .	388
19.1.1.9 LockValue . . . . .	389
19.1.1.10 OTPmode . . . . .	389
19.1.1.11 Reserved0 . . . . .	389
19.1.1.12 Reserved1 . . . . .	389
19.1.1.13 Reserved2 . . . . .	389
19.1.1.14 RevNum . . . . .	389
19.1.1.15 RFU . . . . .	389
19.1.1.16 Selector . . . . .	389
19.1.1.17 SlotConfig . . . . .	390
19.1.1.18 SlotLocked . . . . .	390
19.1.1.19 SN03 . . . . .	390
19.1.1.20 SN47 . . . . .	390
19.1.1.21 SN8 . . . . .	390
19.1.1.22 UserExtra . . . . .	390
19.1.1.23 X509format . . . . .	390
19.2 _atecc608_config Struct Reference . . . . .	390
19.2.1 Field Documentation . . . . .	391
19.2.1.1 AES_Enable . . . . .	391
19.2.1.2 ChipMode . . . . .	391
19.2.1.3 ChipOptions . . . . .	392
19.2.1.4 Counter0 . . . . .	392
19.2.1.5 Counter1 . . . . .	392
19.2.1.6 CountMatch . . . . .	392
19.2.1.7 I2C_Address . . . . .	392
19.2.1.8 I2C_Enable . . . . .	392
19.2.1.9 KdfIvLoc . . . . .	392
19.2.1.10 KdfIvStr . . . . .	392
19.2.1.11 KeyConfig . . . . .	393
19.2.1.12 LockConfig . . . . .	393
19.2.1.13 LockValue . . . . .	393
19.2.1.14 Reserved1 . . . . .	393
19.2.1.15 Reserved2 . . . . .	393
19.2.1.16 Reserved3 . . . . .	393

19.2.1.17 RevNum . . . . .	393
19.2.1.18 SecureBoot . . . . .	393
19.2.1.19 SlotConfig . . . . .	394
19.2.1.20 SlotLocked . . . . .	394
19.2.1.21 SN03 . . . . .	394
19.2.1.22 SN47 . . . . .	394
19.2.1.23 SN8 . . . . .	394
19.2.1.24 UseLock . . . . .	394
19.2.1.25 UserExtra . . . . .	394
19.2.1.26 UserExtraAdd . . . . .	394
19.2.1.27 VolatileKeyPermission . . . . .	395
19.2.1.28 X509format . . . . .	395
19.3 _atsha204a_config Struct Reference . . . . .	395
19.3.1 Field Documentation . . . . .	395
19.3.1.1 ChipMode . . . . .	395
19.3.1.2 Counter . . . . .	396
19.3.1.3 I2C_Address . . . . .	396
19.3.1.4 I2C_Enable . . . . .	396
19.3.1.5 LastKeyUse . . . . .	396
19.3.1.6 LockConfig . . . . .	396
19.3.1.7 LockValue . . . . .	396
19.3.1.8 OTPmode . . . . .	396
19.3.1.9 Reserved0 . . . . .	396
19.3.1.10 Reserved1 . . . . .	397
19.3.1.11 Reserved2 . . . . .	397
19.3.1.12 RevNum . . . . .	397
19.3.1.13 Selector . . . . .	397
19.3.1.14 SlotConfig . . . . .	397
19.3.1.15 SN03 . . . . .	397
19.3.1.16 SN47 . . . . .	397
19.3.1.17 SN8 . . . . .	397
19.3.1.18 UserExtra . . . . .	398
19.4 _pkcs11_mech_table_e Struct Reference . . . . .	398
19.4.1 Field Documentation . . . . .	398
19.4.1.1 info . . . . .	398
19.4.1.2 type . . . . .	398
19.5 _pkcs11_attr_model Struct Reference . . . . .	398
19.5.1 Field Documentation . . . . .	398
19.5.1.1 func . . . . .	398
19.5.1.2 type . . . . .	399
19.6 _pkcs11_lib_ctx Struct Reference . . . . .	399
19.6.1 Detailed Description . . . . .	399

19.6.2 Field Documentation	399
19.6.2.1 config_path	399
19.6.2.2 create_mutex	399
19.6.2.3 destroy_mutex	399
19.6.2.4 initialized	400
19.6.2.5 lock_mutex	400
19.6.2.6 mutex	400
19.6.2.7 slot_cnt	400
19.6.2.8 slots	400
19.6.2.9 unlock_mutex	400
19.7 _pkcs11_object Struct Reference	400
19.7.1 Field Documentation	401
19.7.1.1 attributes	401
19.7.1.2 class_id	401
19.7.1.3 class_type	401
19.7.1.4 config	401
19.7.1.5 count	401
19.7.1.6 data	402
19.7.1.7 flags	402
19.7.1.8 handle_info	402
19.7.1.9 name	402
19.7.1.10 size	402
19.7.1.11 slot	402
19.8 _pkcs11_object_cache_t Struct Reference	402
19.8.1 Field Documentation	403
19.8.1.1 handle	403
19.8.1.2 object	403
19.9 _pkcs11_session_ctx Struct Reference	403
19.9.1 Detailed Description	403
19.9.2 Field Documentation	403
19.9.2.1 active_object	404
19.9.2.2 attrib_count	404
19.9.2.3 attrib_list	404
19.9.2.4 error	404
19.9.2.5 handle	404
19.9.2.6 initialized	404
19.9.2.7 logged_in	404
19.9.2.8 object_count	404
19.9.2.9 object_index	405
19.9.2.10 read_key	405
19.9.2.11 slot	405
19.9.2.12 state	405

19.10 <a href="#">_pkcs11_slot_ctx Struct Reference</a>	405
19.10.1 Detailed Description	405
19.10.2 Field Documentation	406
19.10.2.1 <a href="#">cfg_zone</a>	406
19.10.2.2 <a href="#">device_ctx</a>	406
19.10.2.3 <a href="#">flags</a>	406
19.10.2.4 <a href="#">initialized</a>	406
19.10.2.5 <a href="#">interface_config</a>	406
19.10.2.6 <a href="#">label</a>	406
19.10.2.7 <a href="#">session</a>	406
19.10.2.8 <a href="#">slot_id</a>	407
19.10.2.9 <a href="#">so_pin_handle</a>	407
19.10.2.10 <a href="#">user_pin_handle</a>	407
19.11 <a href="#">atca_aes_cbc_ctx Struct Reference</a>	407
19.11.1 Field Documentation	407
19.11.1.1 <a href="#">ciphertext</a>	407
19.11.1.2 <a href="#">device</a>	408
19.11.1.3 <a href="#">key_block</a>	408
19.11.1.4 <a href="#">key_id</a>	408
19.12 <a href="#">atca_aes_cmac_ctx Struct Reference</a>	408
19.12.1 Field Documentation	408
19.12.1.1 <a href="#">block</a>	408
19.12.1.2 <a href="#">block_size</a>	409
19.12.1.3 <a href="#">cbc_ctx</a>	409
19.13 <a href="#">atca_aes_ctr_ctx Struct Reference</a>	409
19.13.1 Field Documentation	409
19.13.1.1 <a href="#">cb</a>	409
19.13.1.2 <a href="#">counter_size</a>	410
19.13.1.3 <a href="#">device</a>	410
19.13.1.4 <a href="#">key_block</a>	410
19.13.1.5 <a href="#">key_id</a>	410
19.14 <a href="#">atca_aes_gcm_ctx Struct Reference</a>	410
19.14.1 Detailed Description	411
19.14.2 Field Documentation	411
19.14.2.1 <a href="#">aad_size</a>	411
19.14.2.2 <a href="#">cb</a>	411
19.14.2.3 <a href="#">ciphertext_block</a>	412
19.14.2.4 <a href="#">data_size</a>	412
19.14.2.5 <a href="#">enc_cb</a>	412
19.14.2.6 <a href="#">h</a>	412
19.14.2.7 <a href="#">j0</a>	412
19.14.2.8 <a href="#">key_block</a>	412

19.14.2.9 key_id	413
19.14.2.10 partial_aad	413
19.14.2.11 partial_aad_size	413
19.14.2.12 y	413
19.15 atca_check_mac_in_out Struct Reference	413
19.15.1 Detailed Description	414
19.15.2 Field Documentation	414
19.15.2.1 client_chal	414
19.15.2.2 client_resp	414
19.15.2.3 key_id	414
19.15.2.4 mode	414
19.15.2.5 other_data	414
19.15.2.6 otp	415
19.15.2.7 slot_key	415
19.15.2.8 sn	415
19.15.2.9 target_key	415
19.15.2.10 temp_key	415
19.16 atca_command Struct Reference	415
19.16.1 Detailed Description	416
19.16.2 Field Documentation	416
19.16.2.1 clock_divider	416
19.16.2.2 dt	416
19.16.2.3 execution_time_msec	416
19.17 atca_decrypt_in_out Struct Reference	416
19.17.1 Detailed Description	416
19.18 atca_derive_key_in_out Struct Reference	417
19.18.1 Detailed Description	417
19.18.2 Field Documentation	417
19.18.2.1 mode	417
19.18.2.2 parent_key	417
19.18.2.3 sn	418
19.18.2.4 target_key	418
19.18.2.5 target_key_id	418
19.18.2.6 temp_key	418
19.19 atca_derive_key_mac_in_out Struct Reference	418
19.19.1 Detailed Description	419
19.19.2 Field Documentation	419
19.19.2.1 mac	419
19.19.2.2 mode	419
19.19.2.3 parent_key	419
19.19.2.4 sn	419
19.19.2.5 target_key_id	419

19.20 atca_device Struct Reference	420
19.20.1 Detailed Description	420
19.20.2 Field Documentation	420
19.20.2.1 mCommands	420
19.20.2.2 mlface	420
19.20.2.3 session_counter	420
19.20.2.4 session_key	421
19.20.2.5 session_key_id	421
19.20.2.6 session_key_len	421
19.20.2.7 session_state	421
19.21 atca_gen_dig_in_out Struct Reference	421
19.21.1 Detailed Description	422
19.21.2 Field Documentation	422
19.21.2.1 counter	422
19.21.2.2 is_key_nomac	422
19.21.2.3 key_conf	422
19.21.2.4 key_id	422
19.21.2.5 other_data	422
19.21.2.6 slot_conf	423
19.21.2.7 slot_locked	423
19.21.2.8 sn	423
19.21.2.9 stored_value	423
19.21.2.10 temp_key	423
19.21.2.11 zone	423
19.22 atca_gen_key_in_out Struct Reference	424
19.22.1 Detailed Description	424
19.22.2 Field Documentation	424
19.22.2.1 key_id	424
19.22.2.2 mode	424
19.22.2.3 other_data	425
19.22.2.4 public_key	425
19.22.2.5 public_key_size	425
19.22.2.6 sn	425
19.22.2.7 temp_key	425
19.23 atca_hmac_in_out Struct Reference	425
19.23.1 Detailed Description	426
19.24 atca_iface Struct Reference	426
19.24.1 Detailed Description	426
19.24.2 Field Documentation	426
19.24.2.1 atidle	427
19.24.2.2 atinit	427
19.24.2.3 atpostinit	427



19.24.2.4 atreceive . . . . .	427
19.24.2.5 atsend . . . . .	427
19.24.2.6 atsleep . . . . .	427
19.24.2.7 atwake . . . . .	427
19.24.2.8 hal_data . . . . .	427
19.24.2.9 mifaceCFG . . . . .	428
19.24.2.10 mType . . . . .	428
19.25 atca_include_data_in_out Struct Reference . . . . .	428
19.25.1 Detailed Description . . . . .	428
19.25.2 Field Documentation . . . . .	428
19.25.2.1 mode . . . . .	428
19.26 atca_io_decrypt_in_out Struct Reference . . . . .	428
19.26.1 Field Documentation . . . . .	429
19.26.1.1 data . . . . .	429
19.26.1.2 data_size . . . . .	429
19.26.1.3 io_key . . . . .	429
19.26.1.4 out_nonce . . . . .	429
19.27 atca_jwt_t Struct Reference . . . . .	430
19.27.1 Detailed Description . . . . .	430
19.27.2 Field Documentation . . . . .	430
19.27.2.1 buf . . . . .	430
19.27.2.2 buflen . . . . .	430
19.27.2.3 cur . . . . .	430
19.28 atca_mac_in_out Struct Reference . . . . .	430
19.28.1 Detailed Description . . . . .	431
19.29 atca_nonce_in_out Struct Reference . . . . .	431
19.29.1 Detailed Description . . . . .	431
19.30 atca_secureboot_enc_in_out Struct Reference . . . . .	432
19.30.1 Field Documentation . . . . .	432
19.30.1.1 digest . . . . .	432
19.30.1.2 digest_enc . . . . .	432
19.30.1.3 hashed_key . . . . .	432
19.30.1.4 io_key . . . . .	433
19.30.1.5 temp_key . . . . .	433
19.31 atca_secureboot_mac_in_out Struct Reference . . . . .	433
19.31.1 Field Documentation . . . . .	433
19.31.1.1 digest . . . . .	433
19.31.1.2 hashed_key . . . . .	434
19.31.1.3 mac . . . . .	434
19.31.1.4 mode . . . . .	434
19.31.1.5 param2 . . . . .	434
19.31.1.6 secure_boot_config . . . . .	434

19.31.1.7 signature . . . . .	434
19.32 atca_sha256_ctx Struct Reference . . . . .	435
19.32.1 Field Documentation . . . . .	435
19.32.1.1 block . . . . .	435
19.32.1.2 block_size . . . . .	435
19.32.1.3 total_msg_size . . . . .	435
19.33 atca_sign_internal_in_out Struct Reference . . . . .	435
19.33.1 Detailed Description . . . . .	436
19.33.2 Field Documentation . . . . .	436
19.33.2.1 digest . . . . .	436
19.33.2.2 for_invalidate . . . . .	437
19.33.2.3 is_slot_locked . . . . .	437
19.33.2.4 key_config . . . . .	437
19.33.2.5 key_id . . . . .	437
19.33.2.6 message . . . . .	437
19.33.2.7 mode . . . . .	437
19.33.2.8 slot_config . . . . .	438
19.33.2.9 sn . . . . .	438
19.33.2.10 temp_key . . . . .	438
19.33.2.11 update_count . . . . .	438
19.33.2.12 use_flag . . . . .	438
19.33.2.13 verify_other_data . . . . .	438
19.34 atca_temp_key Struct Reference . . . . .	439
19.34.1 Detailed Description . . . . .	439
19.34.2 Field Documentation . . . . .	439
19.34.2.1 gen_dig_data . . . . .	439
19.34.2.2 gen_key_data . . . . .	439
19.34.2.3 is_64 . . . . .	440
19.34.2.4 key_id . . . . .	440
19.34.2.5 no_mac_flag . . . . .	440
19.34.2.6 source_flag . . . . .	440
19.34.2.7 valid . . . . .	440
19.34.2.8 value . . . . .	440
19.35 atca_verify_in_out Struct Reference . . . . .	441
19.35.1 Detailed Description . . . . .	441
19.36 atca_verify_mac Struct Reference . . . . .	441
19.36.1 Field Documentation . . . . .	442
19.36.1.1 io_key . . . . .	442
19.36.1.2 key_id . . . . .	442
19.36.1.3 mac . . . . .	442
19.36.1.4 mode . . . . .	442
19.36.1.5 msg_dig_buf . . . . .	442

19.36.1.6 other_data . . . . .	443
19.36.1.7 signature . . . . .	443
19.36.1.8 sn . . . . .	443
19.36.1.9 temp_key . . . . .	443
19.37 atca_write_mac_in_out Struct Reference . . . . .	443
19.37.1 Detailed Description . . . . .	444
19.37.2 Field Documentation . . . . .	444
19.37.2.1 auth_mac . . . . .	444
19.37.2.2 encrypted_data . . . . .	444
19.37.2.3 input_data . . . . .	444
19.37.2.4 key_id . . . . .	444
19.37.2.5 sn . . . . .	444
19.37.2.6 temp_key . . . . .	445
19.37.2.7 zone . . . . .	445
19.38 atcacert_build_state_s Struct Reference . . . . .	445
19.38.1 Detailed Description . . . . .	445
19.38.2 Field Documentation . . . . .	445
19.38.2.1 cert . . . . .	446
19.38.2.2 cert_def . . . . .	446
19.38.2.3 cert_size . . . . .	446
19.38.2.4 device_sn . . . . .	446
19.38.2.5 is_device_sn . . . . .	446
19.38.2.6 max_cert_size . . . . .	446
19.39 atcacert_cert_element_s Struct Reference . . . . .	447
19.39.1 Detailed Description . . . . .	447
19.39.2 Field Documentation . . . . .	447
19.39.2.1 cert_loc . . . . .	447
19.39.2.2 device_loc . . . . .	447
19.39.2.3 id . . . . .	447
19.39.2.4 transforms . . . . .	448
19.40 atcacert_cert_loc_s Struct Reference . . . . .	448
19.40.1 Detailed Description . . . . .	448
19.40.2 Field Documentation . . . . .	448
19.40.2.1 count . . . . .	448
19.40.2.2 offset . . . . .	448
19.41 atcacert_def_s Struct Reference . . . . .	448
19.41.1 Detailed Description . . . . .	449
19.41.2 Field Documentation . . . . .	449
19.41.2.1 ca_cert_def . . . . .	450
19.41.2.2 cert_elements . . . . .	450
19.41.2.3 cert_elements_count . . . . .	450
19.41.2.4 cert_sn_dev_loc . . . . .	450

19.41.2.5 cert_template . . . . .	450
19.41.2.6 cert_template_size . . . . .	450
19.41.2.7 chain_id . . . . .	451
19.41.2.8 comp_cert_dev_loc . . . . .	451
19.41.2.9 expire_date_format . . . . .	451
19.41.2.10 expire_years . . . . .	451
19.41.2.11 issue_date_format . . . . .	451
19.41.2.12 private_key_slot . . . . .	451
19.41.2.13 public_key_dev_loc . . . . .	452
19.41.2.14 sn_source . . . . .	452
19.41.2.15 std_cert_elements . . . . .	452
19.41.2.16 tbs_cert_loc . . . . .	452
19.41.2.17 template_id . . . . .	452
19.41.2.18 type . . . . .	452
19.42 atcacert_device_loc_s Struct Reference . . . . .	453
19.42.1 Detailed Description . . . . .	453
19.42.2 Field Documentation . . . . .	453
19.42.2.1 count . . . . .	453
19.42.2.2 is_genkey . . . . .	453
19.42.2.3 offset . . . . .	453
19.42.2.4 slot . . . . .	454
19.42.2.5 zone . . . . .	454
19.43 atcacert_tm_utc_s Struct Reference . . . . .	454
19.43.1 Detailed Description . . . . .	454
19.43.2 Field Documentation . . . . .	454
19.43.2.1 tm_hour . . . . .	454
19.43.2.2 tm_mday . . . . .	454
19.43.2.3 tm_min . . . . .	455
19.43.2.4 tm_mon . . . . .	455
19.43.2.5 tm_sec . . . . .	455
19.43.2.6 tm_year . . . . .	455
19.44 ATCAHAL_t Struct Reference . . . . .	455
19.44.1 Detailed Description . . . . .	455
19.44.2 Field Documentation . . . . .	456
19.44.2.1 hal_data . . . . .	456
19.44.2.2 halidle . . . . .	456
19.44.2.3 halinit . . . . .	456
19.44.2.4 halpostinit . . . . .	456
19.44.2.5 halreceive . . . . .	456
19.44.2.6 halrelease . . . . .	456
19.44.2.7 halsend . . . . .	456
19.44.2.8 halsleep . . . . .	457

19.44.2.9 halwake	457
19.45 atcahid Struct Reference	457
19.45.1 Field Documentation	457
19.45.1.1 kits [1/2]	457
19.45.1.2 kits [2/2]	457
19.45.1.3 num_kits_found	457
19.46 atcal2Cmaster Struct Reference	458
19.46.1 Detailed Description	458
19.46.2 Field Documentation	458
19.46.2.1 bus_index	458
19.46.2.2 i2c_file	458
19.46.2.3 id	458
19.46.2.4 ref_ct	458
19.46.2.5 twi_id	459
19.46.2.6 twi_master_instance	459
19.47 ATCAIfaceCfg Struct Reference	459
19.47.1 Field Documentation	460
19.47.1.1 "@1	460
19.47.1.2 atcacustom	460
19.47.1.3 atcahid	460
19.47.1.4 atcai2c	460
19.47.1.5 atcaspi	460
19.47.1.6 atcaswi	461
19.47.1.7 atcauart	461
19.47.1.8 baud	461
19.47.1.9 bus	461
19.47.1.10 cfg_data	461
19.47.1.11 dev_identity	461
19.47.1.12 dev_interface	461
19.47.1.13 devtype	461
19.47.1.14 halidle	462
19.47.1.15 halinit	462
19.47.1.16 halpostinit	462
19.47.1.17 halreceive	462
19.47.1.18 halrelease	462
19.47.1.19 halsend	462
19.47.1.20 halsleep	462
19.47.1.21 halwake	462
19.47.1.22 idx	463
19.47.1.23 iface_type	463
19.47.1.24 packetsize	463
19.47.1.25 parity	463

19.47.1.26 pid . . . . .	463
19.47.1.27 port . . . . .	463
19.47.1.28 rx_retries . . . . .	463
19.47.1.29 select_pin . . . . .	463
19.47.1.30 slave_address . . . . .	464
19.47.1.31 stopbits . . . . .	464
19.47.1.32 vid . . . . .	464
19.47.1.33 wake_delay . . . . .	464
19.47.1.34 wordsize . . . . .	464
19.48 ATCAPacket Struct Reference . . . . .	464
19.48.1 Field Documentation . . . . .	464
19.48.1.1 _reserved . . . . .	465
19.48.1.2 data . . . . .	465
19.48.1.3 execTime . . . . .	465
19.48.1.4 opcode . . . . .	465
19.48.1.5 param1 . . . . .	465
19.48.1.6 param2 . . . . .	465
19.48.1.7 txsize . . . . .	465
19.49 atcaSPImaster Struct Reference . . . . .	465
19.49.1 Field Documentation . . . . .	466
19.49.1.1 ref_ct . . . . .	466
19.49.1.2 spi_file . . . . .	466
19.50 atcaSWImaster Struct Reference . . . . .	466
19.50.1 Detailed Description . . . . .	466
19.50.2 Field Documentation . . . . .	466
19.50.2.1 bus_index . . . . .	467
19.50.2.2 ref_ct . . . . .	467
19.50.2.3 sercom_core_freq . . . . .	467
19.50.2.4 usart_instance . . . . .	467
19.50.2.5 USART_SWI . . . . .	467
19.51 CK_AES_CBC_ENCRYPT_DATA_PARAMS Struct Reference . . . . .	467
19.51.1 Field Documentation . . . . .	467
19.51.1.1 iv . . . . .	468
19.51.1.2 length . . . . .	468
19.51.1.3 pData . . . . .	468
19.52 CK_AES_CCM_PARAMS Struct Reference . . . . .	468
19.52.1 Field Documentation . . . . .	468
19.52.1.1 pAAD . . . . .	468
19.52.1.2 pNonce . . . . .	468
19.52.1.3 ulAADLen . . . . .	469
19.52.1.4 ulDataLen . . . . .	469
19.52.1.5 ulMACLen . . . . .	469

19.52.1.6 ulNonceLen . . . . .	469
19.53 CK_AES_CTR_PARAMS Struct Reference . . . . .	469
19.53.1 Field Documentation . . . . .	469
19.53.1.1 cb . . . . .	469
19.53.1.2 ulCounterBits . . . . .	470
19.54 CK_AES_GCM_PARAMS Struct Reference . . . . .	470
19.54.1 Field Documentation . . . . .	470
19.54.1.1 pAAD . . . . .	470
19.54.1.2 plv . . . . .	470
19.54.1.3 ulAADLen . . . . .	470
19.54.1.4 ullvBits . . . . .	470
19.54.1.5 ullvLen . . . . .	471
19.54.1.6 ulTagBits . . . . .	471
19.55 CK_ARIA_CBC_ENCRYPT_DATA_PARAMS Struct Reference . . . . .	471
19.55.1 Field Documentation . . . . .	471
19.55.1.1 iv . . . . .	471
19.55.1.2 length . . . . .	471
19.55.1.3 pData . . . . .	471
19.56 CK_ATTRIBUTE Struct Reference . . . . .	471
19.56.1 Field Documentation . . . . .	472
19.56.1.1 pValue . . . . .	472
19.56.1.2 type . . . . .	472
19.56.1.3 ulValueLen . . . . .	472
19.57 CK_C_INITIALIZE_ARGS Struct Reference . . . . .	472
19.57.1 Field Documentation . . . . .	472
19.57.1.1 CreateMutex . . . . .	473
19.57.1.2 DestroyMutex . . . . .	473
19.57.1.3 flags . . . . .	473
19.57.1.4 LockMutex . . . . .	473
19.57.1.5 pReserved . . . . .	473
19.57.1.6 UnlockMutex . . . . .	473
19.58 CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS Struct Reference . . . . .	473
19.58.1 Field Documentation . . . . .	474
19.58.1.1 iv . . . . .	474
19.58.1.2 length . . . . .	474
19.58.1.3 pData . . . . .	474
19.59 CK_CAMELLIA_CTR_PARAMS Struct Reference . . . . .	474
19.59.1 Field Documentation . . . . .	474
19.59.1.1 cb . . . . .	474
19.59.1.2 ulCounterBits . . . . .	474
19.60 CK_CCM_PARAMS Struct Reference . . . . .	475
19.60.1 Field Documentation . . . . .	475

19.60.1.1 pAAD . . . . .	475
19.60.1.2 pNonce . . . . .	475
19.60.1.3 ulAADLen . . . . .	475
19.60.1.4 ulDataLen . . . . .	475
19.60.1.5 ulMACLen . . . . .	475
19.60.1.6 ulNonceLen . . . . .	476
19.61 CK_CMS_SIG_PARAMS Struct Reference . . . . .	476
19.61.1 Field Documentation . . . . .	476
19.61.1.1 certificateHandle . . . . .	476
19.61.1.2 pContentType . . . . .	476
19.61.1.3 pDigestMechanism . . . . .	476
19.61.1.4 pRequestedAttributes . . . . .	476
19.61.1.5 pRequiredAttributes . . . . .	477
19.61.1.6 pSigningMechanism . . . . .	477
19.61.1.7 ulRequestedAttributesLen . . . . .	477
19.61.1.8 ulRequiredAttributesLen . . . . .	477
19.62 CK_DATE Struct Reference . . . . .	477
19.62.1 Field Documentation . . . . .	477
19.62.1.1 day . . . . .	477
19.62.1.2 month . . . . .	478
19.62.1.3 year . . . . .	478
19.63 CK_DES_CBC_ENCRYPT_DATA_PARAMS Struct Reference . . . . .	478
19.63.1 Field Documentation . . . . .	478
19.63.1.1 iv . . . . .	478
19.63.1.2 length . . . . .	478
19.63.1.3 pData . . . . .	478
19.64 CK_DSA_PARAMETER_GEN_PARAM Struct Reference . . . . .	478
19.64.1 Field Documentation . . . . .	479
19.64.1.1 hash . . . . .	479
19.64.1.2 pSeed . . . . .	479
19.64.1.3 ulIndex . . . . .	479
19.64.1.4 ulSeedLen . . . . .	479
19.65 CK_ECDH1_DERIVE_PARAMS Struct Reference . . . . .	479
19.65.1 Field Documentation . . . . .	480
19.65.1.1 kdf . . . . .	480
19.65.1.2 pPublicData . . . . .	480
19.65.1.3 pSharedData . . . . .	480
19.65.1.4 ulPublicDataLen . . . . .	480
19.65.1.5 ulSharedDataLen . . . . .	480
19.66 CK_ECDH2_DERIVE_PARAMS Struct Reference . . . . .	480
19.66.1 Field Documentation . . . . .	481
19.66.1.1 hPrivateData . . . . .	481



19.66.1.2 kdf	481
19.66.1.3 pPublicData	481
19.66.1.4 pPublicData2	481
19.66.1.5 pSharedData	481
19.66.1.6 ulPrivateDataLen	481
19.66.1.7 ulPublicDataLen	481
19.66.1.8 ulPublicDataLen2	482
19.66.1.9 ulSharedDataLen	482
19.67 CK_ECDH_AES_KEY_WRAP_PARAMS Struct Reference	482
19.67.1 Field Documentation	482
19.67.1.1 kdf	482
19.67.1.2 pSharedData	482
19.67.1.3 ulAESKeyBits	482
19.67.1.4 ulSharedDataLen	483
19.68 CK_ECMQV_DERIVE_PARAMS Struct Reference	483
19.68.1 Field Documentation	483
19.68.1.1 hPrivateData	483
19.68.1.2 kdf	483
19.68.1.3 pPublicData	483
19.68.1.4 pPublicData2	484
19.68.1.5 pSharedData	484
19.68.1.6 publicKey	484
19.68.1.7 ulPrivateDataLen	484
19.68.1.8 ulPublicDataLen	484
19.68.1.9 ulPublicDataLen2	484
19.68.1.10 ulSharedDataLen	484
19.69 CK_FUNCTION_LIST Struct Reference	484
19.69.1 Field Documentation	485
19.69.1.1 version	485
19.70 CK_GCM_PARAMS Struct Reference	485
19.70.1 Field Documentation	485
19.70.1.1 pAAD	485
19.70.1.2 plv	485
19.70.1.3 ulAADLen	486
19.70.1.4 ullvBits	486
19.70.1.5 ullvLen	486
19.70.1.6 ulTagBits	486
19.71 CK_GOSTR3410_DERIVE_PARAMS Struct Reference	486
19.71.1 Field Documentation	486
19.71.1.1 kdf	486
19.71.1.2 pPublicData	487
19.71.1.3 pUKM	487

19.71.1.4 ulPublicDataLen . . . . .	487
19.71.1.5 ulUKMLen . . . . .	487
19.72 CK_GOSTR3410_KEY_WRAP_PARAMS Struct Reference . . . . .	487
19.72.1 Field Documentation . . . . .	487
19.72.1.1 hKey . . . . .	487
19.72.1.2 pUKM . . . . .	488
19.72.1.3 pWrapOID . . . . .	488
19.72.1.4 ulUKMLen . . . . .	488
19.72.1.5 ulWrapOIDLen . . . . .	488
19.73 CK_INFO Struct Reference . . . . .	488
19.73.1 Field Documentation . . . . .	488
19.73.1.1 cryptokiVersion . . . . .	488
19.73.1.2 flags . . . . .	489
19.73.1.3 libraryDescription . . . . .	489
19.73.1.4 libraryVersion . . . . .	489
19.73.1.5 manufacturerID . . . . .	489
19.74 CK_KEA_DERIVE_PARAMS Struct Reference . . . . .	489
19.74.1 Field Documentation . . . . .	489
19.74.1.1 isSender . . . . .	489
19.74.1.2 pPublicData . . . . .	490
19.74.1.3 pRandomA . . . . .	490
19.74.1.4 pRandomB . . . . .	490
19.74.1.5 ulPublicDataLen . . . . .	490
19.74.1.6 ulRandomLen . . . . .	490
19.75 CK_KEY_DERIVATION_STRING_DATA Struct Reference . . . . .	490
19.75.1 Field Documentation . . . . .	490
19.75.1.1 pData . . . . .	491
19.75.1.2 ulLen . . . . .	491
19.76 CK_KEY_WRAP_SET_OAEP_PARAMS Struct Reference . . . . .	491
19.76.1 Field Documentation . . . . .	491
19.76.1.1 bBC . . . . .	491
19.76.1.2 pX . . . . .	491
19.76.1.3 ulXLen . . . . .	491
19.77 CK_KIP_PARAMS Struct Reference . . . . .	491
19.77.1 Field Documentation . . . . .	492
19.77.1.1 hKey . . . . .	492
19.77.1.2 pMechanism . . . . .	492
19.77.1.3 pSeed . . . . .	492
19.77.1.4 ulSeedLen . . . . .	492
19.78 CK_MECHANISM Struct Reference . . . . .	492
19.78.1 Field Documentation . . . . .	493
19.78.1.1 mechanism . . . . .	493

19.78.1.2 pParameter . . . . .	493
19.78.1.3 ulParameterLen . . . . .	493
19.79 CK_MECHANISM_INFO Struct Reference . . . . .	493
19.79.1 Field Documentation . . . . .	493
19.79.1.1 flags . . . . .	493
19.79.1.2 ulMaxKeySize . . . . .	494
19.79.1.3 ulMinKeySize . . . . .	494
19.80 CK_OTP_PARAM Struct Reference . . . . .	494
19.80.1 Field Documentation . . . . .	494
19.80.1.1 pValue . . . . .	494
19.80.1.2 type . . . . .	494
19.80.1.3 ulValueLen . . . . .	494
19.81 CK_OTP_PARAMS Struct Reference . . . . .	494
19.81.1 Field Documentation . . . . .	495
19.81.1.1 pParams . . . . .	495
19.81.1.2 ulCount . . . . .	495
19.82 CK_OTP_SIGNATURE_INFO Struct Reference . . . . .	495
19.82.1 Field Documentation . . . . .	495
19.82.1.1 pParams . . . . .	495
19.82.1.2 ulCount . . . . .	495
19.83 CK_PBE_PARAMS Struct Reference . . . . .	496
19.83.1 Field Documentation . . . . .	496
19.83.1.1 plnitVector . . . . .	496
19.83.1.2 pPassword . . . . .	496
19.83.1.3 pSalt . . . . .	496
19.83.1.4 ulIteration . . . . .	496
19.83.1.5 ulPasswordLen . . . . .	496
19.83.1.6 ulSaltLen . . . . .	497
19.84 CK_PKCS5_PBKD2_PARAMS Struct Reference . . . . .	497
19.84.1 Field Documentation . . . . .	497
19.84.1.1 iterations . . . . .	497
19.84.1.2 pPassword . . . . .	497
19.84.1.3 pPrfData . . . . .	497
19.84.1.4 prf . . . . .	498
19.84.1.5 pSaltSourceData . . . . .	498
19.84.1.6 saltSource . . . . .	498
19.84.1.7 ulPasswordLen . . . . .	498
19.84.1.8 ulPrfDataLen . . . . .	498
19.84.1.9 ulSaltSourceDataLen . . . . .	498
19.85 CK_PKCS5_PBKD2_PARAMS2 Struct Reference . . . . .	498
19.85.1 Field Documentation . . . . .	499
19.85.1.1 iterations . . . . .	499

19.85.1.2 pPassword	499
19.85.1.3 pPrfData	499
19.85.1.4 prf	499
19.85.1.5 pSaltSourceData	499
19.85.1.6 saltSource	499
19.85.1.7 ulPasswordLen	499
19.85.1.8 ulPrfDataLen	500
19.85.1.9 ulSaltSourceDataLen	500
19.86 CK_RC2_CBC_PARAMS Struct Reference	500
19.86.1 Field Documentation	500
19.86.1.1 iv	500
19.86.1.2 ulEffectiveBits	500
19.87 CK_RC2_MAC_GENERAL_PARAMS Struct Reference	500
19.87.1 Field Documentation	501
19.87.1.1 ulEffectiveBits	501
19.87.1.2 ulMacLength	501
19.88 CK_RC5_CBC_PARAMS Struct Reference	501
19.88.1 Field Documentation	501
19.88.1.1 plv	501
19.88.1.2 ullvLen	501
19.88.1.3 ulRounds	502
19.88.1.4 ulWordsize	502
19.89 CK_RC5_MAC_GENERAL_PARAMS Struct Reference	502
19.89.1 Field Documentation	502
19.89.1.1 ulMacLength	502
19.89.1.2 ulRounds	502
19.89.1.3 ulWordsize	502
19.90 CK_RC5_PARAMS Struct Reference	502
19.90.1 Field Documentation	503
19.90.1.1 ulRounds	503
19.90.1.2 ulWordsize	503
19.91 CK_RSA_AES_KEY_WRAP_PARAMS Struct Reference	503
19.91.1 Field Documentation	503
19.91.1.1 pOAEPParams	503
19.91.1.2 ulAESKeyBits	503
19.92 CK_RSA_PKCS_OAEP_PARAMS Struct Reference	504
19.92.1 Field Documentation	504
19.92.1.1 hashAlg	504
19.92.1.2 mgf	504
19.92.1.3 pSourceData	504
19.92.1.4 source	504
19.92.1.5 ulSourceDataLen	504

19.93 CK_RSA_PKCS_PSS_PARAMS Struct Reference	505
19.93.1 Field Documentation	505
19.93.1.1 hashAlg	505
19.93.1.2 mgf	505
19.93.1.3 sLen	505
19.94 CK_SEED_CBC_ENCRYPT_DATA_PARAMS Struct Reference	505
19.94.1 Field Documentation	505
19.94.1.1 iv	506
19.94.1.2 length	506
19.94.1.3 pData	506
19.95 CK_SESSION_INFO Struct Reference	506
19.95.1 Field Documentation	506
19.95.1.1 flags	506
19.95.1.2 slotID	506
19.95.1.3 state	507
19.95.1.4 ulDeviceError	507
19.96 CK_SKIPJACK_PRIVATE_WRAP_PARAMS Struct Reference	507
19.96.1 Field Documentation	507
19.96.1.1 pBaseG	507
19.96.1.2 pPassword	507
19.96.1.3 pPrimeP	508
19.96.1.4 pPublicData	508
19.96.1.5 pRandomA	508
19.96.1.6 pSubprimeQ	508
19.96.1.7 ulPAndGLen	508
19.96.1.8 ulPasswordLen	508
19.96.1.9 ulPublicDataLen	508
19.96.1.10 ulQLen	508
19.96.1.11 ulRandomLen	509
19.97 CK_SKIPJACK_RELAYX_PARAMS Struct Reference	509
19.97.1 Field Documentation	509
19.97.1.1 pNewPassword	509
19.97.1.2 pNewPublicData	509
19.97.1.3 pNewRandomA	509
19.97.1.4 pOldPassword	510
19.97.1.5 pOldPublicData	510
19.97.1.6 pOldRandomA	510
19.97.1.7 pOldWrappedX	510
19.97.1.8 ulNewPasswordLen	510
19.97.1.9 ulNewPublicDataLen	510
19.97.1.10 ulNewRandomLen	510
19.97.1.11 ulOldPasswordLen	510

19.97.1.12 ulOldPublicDataLen . . . . .	511
19.97.1.13 ulOldRandomLen . . . . .	511
19.97.1.14 ulOldWrappedXLen . . . . .	511
19.98 CK_SLOT_INFO Struct Reference . . . . .	511
19.98.1 Field Documentation . . . . .	511
19.98.1.1 firmwareVersion . . . . .	511
19.98.1.2 flags . . . . .	511
19.98.1.3 hardwareVersion . . . . .	512
19.98.1.4 manufacturerID . . . . .	512
19.98.1.5 slotDescription . . . . .	512
19.99 CK_SSL3_KEY_MAT_OUT Struct Reference . . . . .	512
19.99.1 Field Documentation . . . . .	512
19.99.1.1 hClientKey . . . . .	512
19.99.1.2 hClientMacSecret . . . . .	512
19.99.1.3 hServerKey . . . . .	513
19.99.1.4 hServerMacSecret . . . . .	513
19.99.1.5 pIVClient . . . . .	513
19.99.1.6 pIVServer . . . . .	513
19.100 CK_SSL3_KEY_MAT_PARAMS Struct Reference . . . . .	513
19.100.1 Field Documentation . . . . .	513
19.100.1.1 blsExport . . . . .	513
19.100.1.2 pReturnedKeyMaterial . . . . .	514
19.100.1.3 RandomInfo . . . . .	514
19.100.1.4 ulIVSizeInBits . . . . .	514
19.100.1.5 ulKeySizeInBits . . . . .	514
19.100.1.6 ulMacSizeInBits . . . . .	514
19.101 CK_SSL3_MASTER_KEY_DERIVE_PARAMS Struct Reference . . . . .	514
19.101.1 Field Documentation . . . . .	514
19.101.1.1 pVersion . . . . .	515
19.101.1.2 RandomInfo . . . . .	515
19.102 CK_SSL3_RANDOM_DATA Struct Reference . . . . .	515
19.102.1 Field Documentation . . . . .	515
19.102.1.1 pClientRandom . . . . .	515
19.102.1.2 pServerRandom . . . . .	515
19.102.1.3 ulClientRandomLen . . . . .	515
19.102.1.4 ulServerRandomLen . . . . .	516
19.103 CK_TLS12_KEY_MAT_PARAMS Struct Reference . . . . .	516
19.103.1 Field Documentation . . . . .	516
19.103.1.1 blsExport . . . . .	516
19.103.1.2 pReturnedKeyMaterial . . . . .	516
19.103.1.3 prfHashMechanism . . . . .	516
19.103.1.4 RandomInfo . . . . .	516

19.103.1.5 ulIVSizeInBits . . . . .	517
19.103.1.6 ulKeySizeInBits . . . . .	517
19.103.1.7 ulMacSizeInBits . . . . .	517
19.104 CK_TLS12_MASTER_KEY_DERIVE_PARAMS Struct Reference . . . . .	517
19.104.1 Field Documentation . . . . .	517
19.104.1.1 prfHashMechanism . . . . .	517
19.104.1.2 pVersion . . . . .	517
19.104.1.3 RandomInfo . . . . .	518
19.105 CK_TLS_KDF_PARAMS Struct Reference . . . . .	518
19.105.1 Field Documentation . . . . .	518
19.105.1.1 pContextData . . . . .	518
19.105.1.2 pLabel . . . . .	518
19.105.1.3 prfMechanism . . . . .	518
19.105.1.4 RandomInfo . . . . .	518
19.105.1.5 ulContextDataLength . . . . .	519
19.105.1.6 ulLabelLength . . . . .	519
19.106 CK_TLS_MAC_PARAMS Struct Reference . . . . .	519
19.106.1 Field Documentation . . . . .	519
19.106.1.1 prfHashMechanism . . . . .	519
19.106.1.2 ulMacLength . . . . .	519
19.106.1.3 ulServerOrClient . . . . .	519
19.107 CK_TLS_PRF_PARAMS Struct Reference . . . . .	519
19.107.1 Field Documentation . . . . .	520
19.107.1.1 pLabel . . . . .	520
19.107.1.2 pOutput . . . . .	520
19.107.1.3 pSeed . . . . .	520
19.107.1.4 pulOutputLen . . . . .	520
19.107.1.5 ulLabelLen . . . . .	520
19.107.1.6 ulSeedLen . . . . .	520
19.108 CK_TOKEN_INFO Struct Reference . . . . .	521
19.108.1 Field Documentation . . . . .	521
19.108.1.1 firmwareVersion . . . . .	521
19.108.1.2 flags . . . . .	521
19.108.1.3 hardwareVersion . . . . .	521
19.108.1.4 label . . . . .	522
19.108.1.5 manufacturerID . . . . .	522
19.108.1.6 model . . . . .	522
19.108.1.7 serialNumber . . . . .	522
19.108.1.8 ulFreePrivateMemory . . . . .	522
19.108.1.9 ulFreePublicMemory . . . . .	522
19.108.1.10 ulMaxPinLen . . . . .	522
19.108.1.11 ulMaxRwSessionCount . . . . .	522

19.108.1.12 ulMaxSessionCount . . . . .	523
19.108.1.13 ulMinPinLen . . . . .	523
19.108.1.14 ulRwSessionCount . . . . .	523
19.108.1.15 ulSessionCount . . . . .	523
19.108.1.16 ulTotalPrivateMemory . . . . .	523
19.108.1.17 ulTotalPublicMemory . . . . .	523
19.108.1.18 utcTime . . . . .	523
19.109 CK_VERSION Struct Reference . . . . .	523
19.109.1 Field Documentation . . . . .	524
19.109.1.1 major . . . . .	524
19.109.1.2 minor . . . . .	524
19.110 CK_WTLS_KEY_MAT_OUT Struct Reference . . . . .	524
19.110.1 Field Documentation . . . . .	524
19.110.1.1 hKey . . . . .	524
19.110.1.2 hMacSecret . . . . .	524
19.110.1.3 pIV . . . . .	525
19.111 CK_WTLS_KEY_MAT_PARAMS Struct Reference . . . . .	525
19.111.1 Field Documentation . . . . .	525
19.111.1.1 blsExport . . . . .	525
19.111.1.2 DigestMechanism . . . . .	525
19.111.1.3 pReturnedKeyMaterial . . . . .	525
19.111.1.4 RandomInfo . . . . .	525
19.111.1.5 ulIVSizeInBits . . . . .	526
19.111.1.6 ulKeySizeInBits . . . . .	526
19.111.1.7 ulMacSizeInBits . . . . .	526
19.111.1.8 ulSequenceNumber . . . . .	526
19.112 CK_WTLS_MASTER_KEY_DERIVE_PARAMS Struct Reference . . . . .	526
19.112.1 Field Documentation . . . . .	526
19.112.1.1 DigestMechanism . . . . .	526
19.112.1.2 pVersion . . . . .	527
19.112.1.3 RandomInfo . . . . .	527
19.113 CK_WTLS_PRF_PARAMS Struct Reference . . . . .	527
19.113.1 Field Documentation . . . . .	527
19.113.1.1 DigestMechanism . . . . .	527
19.113.1.2 pLabel . . . . .	527
19.113.1.3 pOutput . . . . .	527
19.113.1.4 pSeed . . . . .	528
19.113.1.5 pulOutputLen . . . . .	528
19.113.1.6 ulLabelLen . . . . .	528
19.113.1.7 ulSeedLen . . . . .	528
19.114 CK_WTLS_RANDOM_DATA Struct Reference . . . . .	528
19.114.1 Field Documentation . . . . .	528



19.114.1.1 pClientRandom . . . . .	528
19.114.1.2 pServerRandom . . . . .	529
19.114.1.3 ulClientRandomLen . . . . .	529
19.114.1.4 ulServerRandomLen . . . . .	529
19.115 CK_X9_42_DH1_DERIVE_PARAMS Struct Reference . . . . .	529
19.115.1 Field Documentation . . . . .	529
19.115.1.1 kdf . . . . .	529
19.115.1.2 pOtherInfo . . . . .	529
19.115.1.3 pPublicData . . . . .	530
19.115.1.4 ulOtherInfoLen . . . . .	530
19.115.1.5 ulPublicDataLen . . . . .	530
19.116 CK_X9_42_DH2_DERIVE_PARAMS Struct Reference . . . . .	530
19.116.1 Field Documentation . . . . .	530
19.116.1.1 hPrivateData . . . . .	530
19.116.1.2 kdf . . . . .	531
19.116.1.3 pOtherInfo . . . . .	531
19.116.1.4 pPublicData . . . . .	531
19.116.1.5 pPublicData2 . . . . .	531
19.116.1.6 ulOtherInfoLen . . . . .	531
19.116.1.7 ulPrivateDataLen . . . . .	531
19.116.1.8 ulPublicDataLen . . . . .	531
19.116.1.9 ulPublicDataLen2 . . . . .	531
19.117 CK_X9_42_MQV_DERIVE_PARAMS Struct Reference . . . . .	532
19.117.1 Field Documentation . . . . .	532
19.117.1.1 hPrivateData . . . . .	532
19.117.1.2 kdf . . . . .	532
19.117.1.3 pOtherInfo . . . . .	532
19.117.1.4 pPublicData . . . . .	532
19.117.1.5 pPublicData2 . . . . .	533
19.117.1.6 publicKey . . . . .	533
19.117.1.7 ulOtherInfoLen . . . . .	533
19.117.1.8 ulPrivateDataLen . . . . .	533
19.117.1.9 ulPublicDataLen . . . . .	533
19.117.1.10 ulPublicDataLen2 . . . . .	533
19.118 CL_HashContext Struct Reference . . . . .	533
19.118.1 Field Documentation . . . . .	534
19.118.1.1 buf . . . . .	534
19.118.1.2 byteCount . . . . .	534
19.118.1.3 byteCountHi . . . . .	534
19.118.1.4 h . . . . .	534
19.119 hid_device Struct Reference . . . . .	534
19.119.1 Field Documentation . . . . .	534

19.119.1.1 read_handle [1/2]	535
19.119.1.2 read_handle [2/2]	535
19.119.1.3 write_handle [1/2]	535
19.119.1.4 write_handle [2/2]	535
19.120 hw_sha256_ctx Struct Reference	535
19.120.1 Field Documentation	535
19.120.1.1 block	535
19.120.1.2 block_size	536
19.120.1.3 total_msg_size	536
19.121 i2c_sam0_instance Struct Reference	536
19.121.1 Field Documentation	536
19.121.1.1 change_baudrate	536
19.121.1.2 i2c_instance	536
19.122 i2c_sam_instance Struct Reference	536
19.122.1 Field Documentation	537
19.122.1.1 change_baudrate	537
19.122.1.2 i2c_instance	537
19.123 i2c_start_instance Struct Reference	537
19.123.1 Field Documentation	537
19.123.1.1 change_baudrate	537
19.123.1.2 i2c_descriptor	537
19.124 memory_parameters Struct Reference	538
19.124.1 Field Documentation	538
19.124.1.1 memory_size	538
19.124.1.2 reserved	538
19.124.1.3 signature	538
19.124.1.4 start_address	538
19.124.1.5 version_info	538
19.125 secure_boot_config_bits Struct Reference	539
19.125.1 Field Documentation	539
19.125.1.1 secure_boot_mode	539
19.125.1.2 secure_boot_persistent_enable	539
19.125.1.3 secure_boot_pub_key	539
19.125.1.4 secure_boot_rand_nonce	539
19.125.1.5 secure_boot_reserved1	539
19.125.1.6 secure_boot_reserved2	540
19.125.1.7 secure_boot_sig_dig	540
19.126 secure_boot_parameters Struct Reference	540
19.126.1 Field Documentation	540
19.126.1.1 app_digest	540
19.126.1.2 memory_params	540
19.126.1.3 s_sha_context	540

19.127 sw_sha256_ctx Struct Reference . . . . .	540
19.127.1 Field Documentation . . . . .	541
19.127.1.1 block . . . . .	541
19.127.1.2 block_size . . . . .	541
19.127.1.3 hash . . . . .	541
19.127.1.4 total_msg_size . . . . .	541
19.128 tng_cert_map_element Struct Reference . . . . .	542
19.128.1 Field Documentation . . . . .	542
19.128.1.1 cert_def . . . . .	542
19.128.1.2 otpcode . . . . .	542
<b>20 File Documentation</b>	<b>543</b>
20.1 api_206a.c File Reference . . . . .	543
20.1.1 Detailed Description . . . . .	544
20.1.2 Function Documentation . . . . .	544
20.1.2.1 sha206a_authenticate() . . . . .	544
20.1.2.2 sha206a_check_dk_useflag_validity() . . . . .	544
20.1.2.3 sha206a_check_pk_useflag_validity() . . . . .	545
20.1.2.4 sha206a_diversify_parent_key() . . . . .	545
20.1.2.5 sha206a_generate_challenge_response_pair() . . . . .	546
20.1.2.6 sha206a_generate_derive_key() . . . . .	546
20.1.2.7 sha206a_get_data_store_lock_status() . . . . .	546
20.1.2.8 sha206a_get_dk_update_count() . . . . .	547
20.1.2.9 sha206a_get_dk_useflag_count() . . . . .	547
20.1.2.10 sha206a_get_pk_useflag_count() . . . . .	547
20.1.2.11 sha206a_read_data_store() . . . . .	548
20.1.2.12 sha206a_verify_device_consumption() . . . . .	548
20.1.2.13 sha206a_write_data_store() . . . . .	549
20.2 api_206a.h File Reference . . . . .	549
20.2.1 Detailed Description . . . . .	550
20.2.2 Macro Definition Documentation . . . . .	550
20.2.2.1 ATCA_SHA206A_DKEY_CONSUMPTION_MASK . . . . .	550
20.2.2.2 ATCA_SHA206A_PKEY_CONSUMPTION_MASK . . . . .	551
20.2.2.3 ATCA_SHA206A_SYMMETRIC_KEY_ID_SLOT . . . . .	551
20.2.2.4 ATCA_SHA206A_ZONE_WRITE_LOCK . . . . .	551
20.2.3 Enumeration Type Documentation . . . . .	551
20.2.3.1 anonymous enum . . . . .	551
20.2.4 Function Documentation . . . . .	551
20.2.4.1 sha206a_authenticate() . . . . .	551
20.2.4.2 sha206a_check_dk_useflag_validity() . . . . .	552
20.2.4.3 sha206a_check_pk_useflag_validity() . . . . .	552
20.2.4.4 sha206a_diversify_parent_key() . . . . .	553

20.2.4.5 sha206a_generate_challenge_response_pair()	553
20.2.4.6 sha206a_generate_derive_key()	553
20.2.4.7 sha206a_get_data_store_lock_status()	554
20.2.4.8 sha206a_get_dk_update_count()	554
20.2.4.9 sha206a_get_dk_useflag_count()	555
20.2.4.10 sha206a_get_pk_useflag_count()	555
20.2.4.11 sha206a_read_data_store()	555
20.2.4.12 sha206a_verify_device_consumption()	556
20.2.4.13 sha206a_write_data_store()	556
20.3 atca_basic.c File Reference	557
20.3.1 Detailed Description	563
20.3.2 Variable Documentation	563
20.3.2.1 _gDevice	563
20.3.2.2 atca_version	563
20.4 atca_basic.h File Reference	563
20.4.1 Detailed Description	570
20.5 atca_bool.h File Reference	571
20.5.1 Detailed Description	571
20.6 atca_cfgs.c File Reference	571
20.6.1 Detailed Description	571
20.7 atca_cfgs.h File Reference	571
20.7.1 Detailed Description	572
20.7.2 Variable Documentation	572
20.7.2.1 cfg_ateccx08a_i2c_default	572
20.7.2.2 cfg_ateccx08a_kitcdc_default	572
20.7.2.3 cfg_ateccx08a_kithid_default	573
20.7.2.4 cfg_ateccx08a_swi_default	573
20.7.2.5 cfg_atsha20xa_i2c_default	573
20.7.2.6 cfg_atsha20xa_kitcdc_default	573
20.7.2.7 cfg_atsha20xa_kithid_default	573
20.7.2.8 cfg_atsha20xa_swi_default	573
20.8 atca_command.c File Reference	574
20.8.1 Detailed Description	574
20.9 atca_command.h File Reference	574
20.9.1 Detailed Description	575
20.10 atca_compiler.h File Reference	575
20.10.1 Detailed Description	575
20.11 atca_crypto_hw_aes.h File Reference	575
20.11.1 Detailed Description	576
20.11.2 Typedef Documentation	576
20.11.2.1 atca_aes_cbc_ctx_t	576
20.11.2.2 atca_aes_cmac_ctx_t	576

20.11.2.3 atca_aes_ctr_ctx_t . . . . .	576
20.12 atca_crypto_hw_aes_cbc.c File Reference . . . . .	576
20.12.1 Detailed Description . . . . .	577
20.13 atca_crypto_hw_aes_cmac.c File Reference . . . . .	577
20.13.1 Detailed Description . . . . .	578
20.14 atca_crypto_hw_aes_ctr.c File Reference . . . . .	578
20.14.1 Detailed Description . . . . .	579
20.15 atca_crypto_sw.h File Reference . . . . .	579
20.15.1 Detailed Description . . . . .	580
20.15.2 Macro Definition Documentation . . . . .	580
20.15.2.1 ATCA_SHA1_DIGEST_SIZE . . . . .	580
20.15.2.2 ATCA_SHA2_256_BLOCK_SIZE . . . . .	581
20.15.2.3 ATCA_SHA2_256_DIGEST_SIZE . . . . .	581
20.15.2.4 MBEDTLS_CMAC_C . . . . .	581
20.15.3 Typedef Documentation . . . . .	581
20.15.3.1 atcac_aes_cmac_ctx . . . . .	581
20.15.3.2 atcac_aes_gcm_ctx . . . . .	581
20.15.3.3 atcac_hmac_sha256_ctx . . . . .	581
20.15.3.4 atcac_sha1_ctx . . . . .	581
20.15.3.5 atcac_sha2_256_ctx . . . . .	582
20.15.4 Function Documentation . . . . .	582
20.15.4.1 atcac_aes_cmac_finish() . . . . .	582
20.15.4.2 atcac_aes_cmac_init() . . . . .	582
20.15.4.3 atcac_aes_cmac_update() . . . . .	582
20.15.4.4 atcac_aes_gcm_aad_update() . . . . .	583
20.15.4.5 atcac_aes_gcm_decrypt_finish() . . . . .	583
20.15.4.6 atcac_aes_gcm_decrypt_start() . . . . .	583
20.15.4.7 atcac_aes_gcm_decrypt_update() . . . . .	584
20.15.4.8 atcac_aes_gcm_encrypt_finish() . . . . .	584
20.15.4.9 atcac_aes_gcm_encrypt_start() . . . . .	584
20.15.4.10 atcac_aes_gcm_encrypt_update() . . . . .	585
20.16 atca_crypto_sw_ecdsa.c File Reference . . . . .	585
20.16.1 Detailed Description . . . . .	585
20.17 atca_crypto_sw_ecdsa.h File Reference . . . . .	585
20.17.1 Detailed Description . . . . .	586
20.18 atca_crypto_sw_rand.c File Reference . . . . .	586
20.18.1 Detailed Description . . . . .	586
20.19 atca_crypto_sw_rand.h File Reference . . . . .	586
20.19.1 Detailed Description . . . . .	587
20.20 atca_crypto_sw_sha1.c File Reference . . . . .	587
20.20.1 Detailed Description . . . . .	587
20.21 atca_crypto_sw_sha1.h File Reference . . . . .	587

20.21.1 Detailed Description . . . . .	588
20.22 atca_crypto_sw_sha2.c File Reference . . . . .	588
20.22.1 Detailed Description . . . . .	588
20.23 atca_crypto_sw_sha2.h File Reference . . . . .	588
20.23.1 Detailed Description . . . . .	589
20.24 atca_debug.c File Reference . . . . .	589
20.24.1 Detailed Description . . . . .	590
20.24.2 Function Documentation . . . . .	590
20.24.2.1 atca_trace() . . . . .	590
20.24.2.2 atca_trace_config() . . . . .	590
20.24.2.3 atca_trace_msg() . . . . .	590
20.24.3 Variable Documentation . . . . .	590
20.24.3.1 g_trace_fp . . . . .	590
20.25 atca_debug.h File Reference . . . . .	590
20.25.1 Function Documentation . . . . .	591
20.25.1.1 atca_trace() . . . . .	591
20.25.1.2 atca_trace_config() . . . . .	591
20.25.1.3 atca_trace_msg() . . . . .	591
20.26 atca_device.c File Reference . . . . .	591
20.26.1 Detailed Description . . . . .	592
20.27 atca_device.h File Reference . . . . .	592
20.27.1 Detailed Description . . . . .	595
20.28 atca_devtypes.h File Reference . . . . .	595
20.28.1 Detailed Description . . . . .	595
20.29 atca_hal.c File Reference . . . . .	596
20.29.1 Detailed Description . . . . .	596
20.30 atca_hal.h File Reference . . . . .	596
20.30.1 Detailed Description . . . . .	597
20.31 atca_helpers.c File Reference . . . . .	597
20.31.1 Detailed Description . . . . .	599
20.31.2 Macro Definition Documentation . . . . .	599
20.31.2.1 B64_IS_EQUAL . . . . .	599
20.31.2.2 B64_IS_INVALID . . . . .	599
20.31.3 Function Documentation . . . . .	599
20.31.3.1 atcab_base64decode() . . . . .	599
20.31.3.2 atcab_base64decode_() . . . . .	600
20.31.3.3 atcab_base64encode() . . . . .	600
20.31.3.4 atcab_base64encode_() . . . . .	601
20.31.3.5 atcab_bin2hex() . . . . .	601
20.31.3.6 atcab_bin2hex_() . . . . .	602
20.31.3.7 atcab_hex2bin() . . . . .	602
20.31.3.8 atcab_hex2bin_() . . . . .	603

20.31.3.9 atcab_memset_s()	603
20.31.3.10 atcab_reversal()	603
20.31.3.11 base64Char()	603
20.31.3.12 base64Index()	605
20.31.3.13 isAlpha()	605
20.31.3.14 isBase64()	606
20.31.3.15 isBase64Digit()	606
20.31.3.16 isDigit()	606
20.31.3.17 isHex()	607
20.31.3.18 isHexAlpha()	607
20.31.3.19 isHexDigit()	607
20.31.3.20 isWhiteSpace()	608
20.31.3.21 packHex()	608
20.31.4 Variable Documentation	609
20.31.4.1 atcab_b64rules_default	609
20.31.4.2 atcab_b64rules_mime	609
20.31.4.3 atcab_b64rules_urlsafe	609
20.32 atca_helpers.h File Reference	609
20.32.1 Detailed Description	610
20.32.2 Function Documentation	610
20.32.2.1 atcab_base64decode()	611
20.32.2.2 atcab_base64decode_()	611
20.32.2.3 atcab_base64encode()	611
20.32.2.4 atcab_base64encode_()	612
20.32.2.5 atcab_bin2hex()	612
20.32.2.6 atcab_bin2hex_()	613
20.32.2.7 atcab_hex2bin()	613
20.32.2.8 atcab_hex2bin_()	614
20.32.2.9 atcab_memset_s()	614
20.32.2.10 atcab_printbin_label()	614
20.32.2.11 atcab_printbin_sp()	614
20.32.2.12 atcab_reversal()	614
20.32.2.13 base64Char()	615
20.32.2.14 base64Index()	615
20.32.2.15 isAlpha()	616
20.32.2.16 isBase64()	616
20.32.2.17 isBase64Digit()	616
20.32.2.18 isDigit()	617
20.32.2.19 isHex()	617
20.32.2.20 isHexAlpha()	618
20.32.2.21 isHexDigit()	618
20.32.2.22 isWhiteSpace()	618

20.32.2.23 packHex()	619
20.32.3 Variable Documentation	619
20.32.3.1 atcab_b64rules_default	619
20.32.3.2 atcab_b64rules_mime	619
20.32.3.3 atcab_b64rules_urlsafe	619
20.33 atca_host.c File Reference	620
20.33.1 Detailed Description	621
20.34 atca_host.h File Reference	621
20.34.1 Detailed Description	624
20.35 atca_iface.c File Reference	624
20.35.1 Detailed Description	625
20.36 atca_iface.h File Reference	625
20.36.1 Detailed Description	626
20.37 atca_jwt.c File Reference	627
20.37.1 Detailed Description	627
20.38 atca_jwt.h File Reference	627
20.38.1 Detailed Description	628
20.39 atca_mbedtls_ecdh.c File Reference	628
20.40 atca_mbedtls_ecdsa.c File Reference	628
20.41 atca_mbedtls_wrap.c File Reference	628
20.41.1 Detailed Description	630
20.41.2 Macro Definition Documentation	630
20.41.2.1 mbedtls_calloc	630
20.41.2.2 mbedtls_free	630
20.41.3 Function Documentation	630
20.41.3.1 atca_mbedtls_cert_add()	630
20.41.3.2 atcac_aes_cmac_finish()	631
20.41.3.3 atcac_aes_cmac_init()	631
20.41.3.4 atcac_aes_cmac_update()	632
20.41.3.5 atcac_aes_gcm_aad_update()	632
20.41.3.6 atcac_aes_gcm_decrypt_finish()	632
20.41.3.7 atcac_aes_gcm_decrypt_start()	633
20.41.3.8 atcac_aes_gcm_decrypt_update()	633
20.41.3.9 atcac_aes_gcm_encrypt_finish()	634
20.41.3.10 atcac_aes_gcm_encrypt_start()	634
20.41.3.11 atcac_aes_gcm_encrypt_update()	635
20.41.3.12 atcac_sw_sha1_finish()	635
20.41.3.13 atcac_sw_sha2_256_finish()	636
20.42 atca_mbedtls_wrap.h File Reference	636
20.43 atca_openssl_interface.c File Reference	636
20.43.1 Detailed Description	638
20.43.2 Function Documentation	638



20.43.2.1 atcac_aes_cmac_finish()	638
20.43.2.2 atcac_aes_cmac_init()	638
20.43.2.3 atcac_aes_cmac_update()	639
20.43.2.4 atcac_aes_gcm_aad_update()	639
20.43.2.5 atcac_aes_gcm_decrypt_finish()	640
20.43.2.6 atcac_aes_gcm_decrypt_start()	640
20.43.2.7 atcac_aes_gcm_decrypt_update()	641
20.43.2.8 atcac_aes_gcm_encrypt_finish()	641
20.43.2.9 atcac_aes_gcm_encrypt_start()	641
20.43.2.10 atcac_aes_gcm_encrypt_update()	642
20.43.2.11 atcac_sw_sha1_finish()	642
20.43.2.12 atcac_sw_sha2_256_finish()	643
20.44 atca_start_config.h File Reference	643
20.45 atca_start_iface.h File Reference	643
20.46 atca_status.h File Reference	643
20.46.1 Detailed Description	644
20.46.2 Macro Definition Documentation	644
20.46.2.1 ATCA_STATUS_AUTH_BIT	644
20.46.3 Enumeration Type Documentation	644
20.46.3.1 ATCA_STATUS	644
20.47 atca_version.h File Reference	646
20.47.1 Detailed Description	646
20.47.2 Macro Definition Documentation	646
20.47.2.1 ATCA_LIBRARY_VERSION_BUILD	646
20.47.2.2 ATCA_LIBRARY_VERSION_DATE	646
20.47.2.3 ATCA_LIBRARY_VERSION_MAJOR	647
20.47.2.4 ATCA_LIBRARY_VERSION_MINOR	647
20.48 atca_wolfssl_interface.c File Reference	647
20.48.1 Detailed Description	647
20.49 atcacert.h File Reference	647
20.49.1 Detailed Description	648
20.50 atcacert_client.c File Reference	648
20.50.1 Detailed Description	649
20.51 atcacert_client.h File Reference	649
20.51.1 Detailed Description	650
20.52 atcacert_date.c File Reference	650
20.52.1 Detailed Description	651
20.53 atcacert_date.h File Reference	651
20.53.1 Detailed Description	653
20.54 atcacert_def.c File Reference	653
20.54.1 Detailed Description	655
20.54.2 Macro Definition Documentation	655

20.54.2.1 ATCACERT_MAX	656
20.54.2.2 ATCACERT_MIN	656
20.55 atcacert_def.h File Reference	656
20.55.1 Detailed Description	659
20.55.2 Macro Definition Documentation	659
20.55.2.1 ATCA_MAX_TRANSFORMS	659
20.56 atcacert_der.c File Reference	660
20.56.1 Detailed Description	660
20.57 atcacert_der.h File Reference	660
20.57.1 Detailed Description	661
20.58 atcacert_host_hw.c File Reference	661
20.58.1 Detailed Description	662
20.59 atcacert_host_hw.h File Reference	662
20.59.1 Detailed Description	662
20.60 atcacert_host_sw.c File Reference	662
20.60.1 Detailed Description	663
20.61 atcacert_host_sw.h File Reference	663
20.61.1 Detailed Description	663
20.62 atcacert_pem.c File Reference	664
20.62.1 Detailed Description	664
20.62.2 Function Documentation	664
20.62.2.1 atcacert_decode_pem()	664
20.62.2.2 atcacert_decode_pem_cert()	665
20.62.2.3 atcacert_decode_pem_csr()	665
20.62.2.4 atcacert_encode_pem()	666
20.62.2.5 atcacert_encode_pem_cert()	666
20.62.2.6 atcacert_encode_pem_csr()	667
20.63 atcacert_pem.h File Reference	667
20.63.1 Detailed Description	668
20.63.2 Macro Definition Documentation	668
20.63.2.1 PEM_CERT_BEGIN	668
20.63.2.2 PEM_CERT_END	668
20.63.2.3 PEM_CSR_BEGIN	669
20.63.2.4 PEM_CSR_END	669
20.63.3 Function Documentation	669
20.63.3.1 atcacert_decode_pem()	669
20.63.3.2 atcacert_decode_pem_cert()	669
20.63.3.3 atcacert_decode_pem_csr()	670
20.63.3.4 atcacert_encode_pem()	670
20.63.3.5 atcacert_encode_pem_cert()	671
20.63.3.6 atcacert_encode_pem_csr()	671
20.64 calib_aes.c File Reference	672

20.64.1 Detailed Description . . . . .	672
20.65 calib_aes_gcm.c File Reference . . . . .	673
20.65.1 Detailed Description . . . . .	673
20.65.2 Macro Definition Documentation . . . . .	674
20.65.2.1 RETURN . . . . .	674
20.65.3 Function Documentation . . . . .	674
20.65.3.1 calib_aes_gcm_aad_update() . . . . .	674
20.65.3.2 calib_aes_gcm_decrypt_finish() . . . . .	674
20.65.3.3 calib_aes_gcm_decrypt_update() . . . . .	675
20.65.3.4 calib_aes_gcm_encrypt_finish() . . . . .	675
20.65.3.5 calib_aes_gcm_encrypt_update() . . . . .	676
20.65.3.6 calib_aes_gcm_init() . . . . .	676
20.65.3.7 calib_aes_gcm_init_rand() . . . . .	677
20.66 calib_aes_gcm.h File Reference . . . . .	677
20.66.1 Detailed Description . . . . .	678
20.67 calib_basic.c File Reference . . . . .	678
20.67.1 Detailed Description . . . . .	679
20.67.2 Macro Definition Documentation . . . . .	679
20.67.2.1 MAX_BUSES . . . . .	679
20.68 calib_basic.h File Reference . . . . .	679
20.69 calib_checkmac.c File Reference . . . . .	685
20.69.1 Detailed Description . . . . .	685
20.70 calib_command.c File Reference . . . . .	685
20.70.1 Detailed Description . . . . .	687
20.70.2 Function Documentation . . . . .	687
20.70.2.1 atAES() . . . . .	687
20.70.2.2 atCalcCrc() . . . . .	688
20.70.2.3 atCheckCrc() . . . . .	688
20.70.2.4 atCheckMAC() . . . . .	688
20.70.2.5 atCounter() . . . . .	689
20.70.2.6 atCRC() . . . . .	689
20.70.2.7 atDeriveKey() . . . . .	689
20.70.2.8 atECDH() . . . . .	690
20.70.2.9 atGenDig() . . . . .	690
20.70.2.10 atGenKey() . . . . .	690
20.70.2.11 atHMAC() . . . . .	691
20.70.2.12 atInfo() . . . . .	691
20.70.2.13 atIsECCFamily() . . . . .	692
20.70.2.14 atIsSHAFamily() . . . . .	692
20.70.2.15 atKDF() . . . . .	692
20.70.2.16 atLock() . . . . .	693
20.70.2.17 atMAC() . . . . .	693

20.70.2.18 atNonce()	693
20.70.2.19 atPause()	694
20.70.2.20 atPrivWrite()	694
20.70.2.21 atRandom()	695
20.70.2.22 atRead()	695
20.70.2.23 atSecureBoot()	695
20.70.2.24 atSelfTest()	696
20.70.2.25 atSHA()	696
20.70.2.26 atSign()	696
20.70.2.27 atUpdateExtra()	697
20.70.2.28 atVerify()	697
20.70.2.29 atWrite()	698
20.70.2.30 isATCAError()	698
20.71 calib_command.h File Reference	698
20.71.1 Detailed Description	716
20.71.2 Macro Definition Documentation	717
20.71.2.1 AES_COUNT	717
20.71.2.2 AES_DATA_SIZE	717
20.71.2.3 AES_INPUT_IDX	717
20.71.2.4 AES_KEYID_IDX	717
20.71.2.5 AES_MODE_DECRYPT	717
20.71.2.6 AES_MODE_ENCRYPT	718
20.71.2.7 AES_MODE_GFM	718
20.71.2.8 AES_MODE_IDX	718
20.71.2.9 AES_MODE_KEY_BLOCK_MASK	718
20.71.2.10 AES_MODE_KEY_BLOCK_POS	718
20.71.2.11 AES_MODE_MASK	718
20.71.2.12 AES_MODE_OP_MASK	719
20.71.2.13 AES_RSP_SIZE	719
20.71.2.14 ATCA_ADDRESS_MASK	719
20.71.2.15 ATCA_ADDRESS_MASK_CONFIG	719
20.71.2.16 ATCA_ADDRESS_MASK_OTP	719
20.71.2.17 ATCA_AES	719
20.71.2.18 ATCA_AES_GFM_SIZE	720
20.71.2.19 ATCA_AES_KEY_TYPE	720
20.71.2.20 ATCA_B283_KEY_TYPE	720
20.71.2.21 ATCA_BLOCK_SIZE	720
20.71.2.22 ATCA_CHECKMAC	720
20.71.2.23 ATCA_CHIPMODE_CLOCK_DIV_M0	720
20.71.2.24 ATCA_CHIPMODE_CLOCK_DIV_M1	721
20.71.2.25 ATCA_CHIPMODE_CLOCK_DIV_M2	721
20.71.2.26 ATCA_CHIPMODE_CLOCK_DIV_MASK	721

20.71.2.27 ATCA_CHIPMODE_I2C_ADDRESS_FLAG . . . . .	721
20.71.2.28 ATCA_CHIPMODE_OFFSET . . . . .	721
20.71.2.29 ATCA_CHIPMODE_TTL_ENABLE_FLAG . . . . .	721
20.71.2.30 ATCA_CHIPMODE_WATCHDOG_LONG . . . . .	722
20.71.2.31 ATCA_CHIPMODE_WATCHDOG_MASK . . . . .	722
20.71.2.32 ATCA_CHIPMODE_WATCHDOG_SHORT . . . . .	722
20.71.2.33 ATCA_CMD_SIZE_MAX . . . . .	722
20.71.2.34 ATCA_CMD_SIZE_MIN . . . . .	722
20.71.2.35 ATCA_COUNT_IDX . . . . .	722
20.71.2.36 ATCA_COUNT_SIZE . . . . .	723
20.71.2.37 ATCA_COUNTER . . . . .	723
20.71.2.38 ATCA_CRC_SIZE . . . . .	723
20.71.2.39 ATCA_DATA_IDX . . . . .	723
20.71.2.40 ATCA_DATA_SIZE . . . . .	723
20.71.2.41 ATCA_DERIVE_KEY . . . . .	723
20.71.2.42 ATCA_ECC_CONFIG_SIZE . . . . .	724
20.71.2.43 ATCA_ECDH . . . . .	724
20.71.2.44 ATCA_GENDIG . . . . .	724
20.71.2.45 ATCA_GENKEY . . . . .	724
20.71.2.46 ATCA_HMAC . . . . .	724
20.71.2.47 ATCA_INFO . . . . .	724
20.71.2.48 ATCA_K283_KEY_TYPE . . . . .	725
20.71.2.49 ATCA_KDF . . . . .	725
20.71.2.50 ATCA_KEY_COUNT . . . . .	725
20.71.2.51 ATCA_KEY_ID_MAX . . . . .	725
20.71.2.52 ATCA_KEY_SIZE . . . . .	725
20.71.2.53 ATCA_LOCK . . . . .	725
20.71.2.54 ATCA_LOCKED . . . . .	726
20.71.2.55 ATCA_MAC . . . . .	726
20.71.2.56 ATCA_NONCE . . . . .	726
20.71.2.57 ATCA_OPCODE_IDX . . . . .	726
20.71.2.58 ATCA_OTP_BLOCK_MAX . . . . .	726
20.71.2.59 ATCA_OTP_SIZE . . . . .	726
20.71.2.60 ATCA_P256_KEY_TYPE . . . . .	727
20.71.2.61 ATCA_PACKET_OVERHEAD . . . . .	727
20.71.2.62 ATCA_PARAM1_IDX . . . . .	727
20.71.2.63 ATCA_PARAM2_IDX . . . . .	727
20.71.2.64 ATCA_PAUSE . . . . .	727
20.71.2.65 ATCA_PRIV_KEY_SIZE . . . . .	727
20.71.2.66 ATCA_PRIVWRITE . . . . .	728
20.71.2.67 ATCA_PUB_KEY_PAD . . . . .	728
20.71.2.68 ATCA_PUB_KEY_SIZE . . . . .	728

20.71.2.69 ATCA_RANDOM . . . . .	728
20.71.2.70 ATCA_READ . . . . .	728
20.71.2.71 ATCA_RSP_DATA_IDX . . . . .	728
20.71.2.72 ATCA_RSP_SIZE_16 . . . . .	729
20.71.2.73 ATCA_RSP_SIZE_32 . . . . .	729
20.71.2.74 ATCA_RSP_SIZE_4 . . . . .	729
20.71.2.75 ATCA_RSP_SIZE_64 . . . . .	729
20.71.2.76 ATCA_RSP_SIZE_72 . . . . .	729
20.71.2.77 ATCA_RSP_SIZE_MAX . . . . .	729
20.71.2.78 ATCA_RSP_SIZE_MIN . . . . .	730
20.71.2.79 ATCA_RSP_SIZE_VAL . . . . .	730
20.71.2.80 ATCA_SECUREBOOT . . . . .	730
20.71.2.81 ATCA_SELFTEST . . . . .	730
20.71.2.82 ATCA_SERIAL_NUM_SIZE . . . . .	730
20.71.2.83 ATCA_SHA . . . . .	730
20.71.2.84 ATCA_SHA_CONFIG_SIZE . . . . .	731
20.71.2.85 ATCA_SHA_DIGEST_SIZE . . . . .	731
20.71.2.86 ATCA_SHA_KEY_TYPE . . . . .	731
20.71.2.87 ATCA_SIG_SIZE . . . . .	731
20.71.2.88 ATCA_SIGN . . . . .	731
20.71.2.89 ATCA_TEMPKEY_KEYID . . . . .	731
20.71.2.90 ATCA_UNLOCKED . . . . .	732
20.71.2.91 ATCA_UPDATE_EXTRA . . . . .	732
20.71.2.92 ATCA_VERIFY . . . . .	732
20.71.2.93 ATCA_WORD_SIZE . . . . .	732
20.71.2.94 ATCA_WRITE . . . . .	732
20.71.2.95 ATCA_ZONE_ENCRYPTED . . . . .	732
20.71.2.96 ATCA_ZONE_MASK . . . . .	733
20.71.2.97 ATCA_ZONE_READWRITE_32 . . . . .	733
20.71.2.98 CHECKMAC_CLIENT_CHALLENGE_IDX . . . . .	733
20.71.2.99 CHECKMAC_CLIENT_CHALLENGE_SIZE . . . . .	733
20.71.2.100 CHECKMAC_CLIENT_COMMAND_SIZE . . . . .	733
20.71.2.101 CHECKMAC_CLIENT_RESPONSE_IDX . . . . .	733
20.71.2.102 CHECKMAC_CLIENT_RESPONSE_SIZE . . . . .	734
20.71.2.103 CHECKMAC_CMD_MATCH . . . . .	734
20.71.2.104 CHECKMAC_CMD_MISMATCH . . . . .	734
20.71.2.105 CHECKMAC_COUNT . . . . .	734
20.71.2.106 CHECKMAC_DATA_IDX . . . . .	734
20.71.2.107 CHECKMAC_KEYID_IDX . . . . .	734
20.71.2.108 CHECKMAC_MODE_BLOCK1_TEMPKEY . . . . .	735
20.71.2.109 CHECKMAC_MODE_BLOCK2_TEMPKEY . . . . .	735
20.71.2.110 CHECKMAC_MODE_CHALLENGE . . . . .	735

20.71.2.111 CHECKMAC_MODE_IDX . . . . .	735
20.71.2.112 CHECKMAC_MODE_INCLUDE_OTP_64 . . . . .	735
20.71.2.113 CHECKMAC_MODE_MASK . . . . .	735
20.71.2.114 CHECKMAC_MODE_SOURCE_FLAG_MATCH . . . . .	736
20.71.2.115 CHECKMAC_OTHER_DATA_SIZE . . . . .	736
20.71.2.116 CHECKMAC_RSP_SIZE . . . . .	736
20.71.2.117 CMD_STATUS_BYTE_COMM . . . . .	736
20.71.2.118 CMD_STATUS_BYTE_ECC . . . . .	736
20.71.2.119 CMD_STATUS_BYTE_EXEC . . . . .	736
20.71.2.120 CMD_STATUS_BYTE_PARSE . . . . .	737
20.71.2.121 CMD_STATUS_SUCCESS . . . . .	737
20.71.2.122 CMD_STATUS_WAKEUP . . . . .	737
20.71.2.123 COUNTER_COUNT . . . . .	737
20.71.2.124 COUNTER_KEYID_IDX . . . . .	737
20.71.2.125 COUNTER_MAX_VALUE . . . . .	737
20.71.2.126 COUNTER_MODE_IDX . . . . .	738
20.71.2.127 COUNTER_MODE_INCREMENT . . . . .	738
20.71.2.128 COUNTER_MODE_MASK . . . . .	738
20.71.2.129 COUNTER_MODE_READ . . . . .	738
20.71.2.130 COUNTER_RSP_SIZE . . . . .	738
20.71.2.131 COUNTER_SIZE . . . . .	738
20.71.2.132 DERIVE_KEY_COUNT_LARGE . . . . .	739
20.71.2.133 DERIVE_KEY_COUNT_SMALL . . . . .	739
20.71.2.134 DERIVE_KEY_MAC_IDX . . . . .	739
20.71.2.135 DERIVE_KEY_MAC_SIZE . . . . .	739
20.71.2.136 DERIVE_KEY_MODE . . . . .	739
20.71.2.137 DERIVE_KEY_RANDOM_FLAG . . . . .	739
20.71.2.138 DERIVE_KEY_RANDOM_IDX . . . . .	740
20.71.2.139 DERIVE_KEY_RSP_SIZE . . . . .	740
20.71.2.140 DERIVE_KEY_TARGETKEY_IDX . . . . .	740
20.71.2.141 ECDH_COUNT . . . . .	740
20.71.2.142 ECDH_KEY_SIZE . . . . .	740
20.71.2.143 ECDH_MODE_COPY_COMPATIBLE . . . . .	740
20.71.2.144 ECDH_MODE_COPY_EEPROM_SLOT . . . . .	740
20.71.2.145 ECDH_MODE_COPY_MASK . . . . .	741
20.71.2.146 ECDH_MODE_COPY_OUTPUT_BUFFER . . . . .	741
20.71.2.147 ECDH_MODE_COPY_TEMP_KEY . . . . .	741
20.71.2.148 ECDH_MODE_OUTPUT_CLEAR . . . . .	741
20.71.2.149 ECDH_MODE_OUTPUT_ENC . . . . .	741
20.71.2.150 ECDH_MODE_OUTPUT_MASK . . . . .	741
20.71.2.151 ECDH_MODE_SOURCE_EEPROM_SLOT . . . . .	741
20.71.2.152 ECDH_MODE_SOURCE_MASK . . . . .	741

20.71.2.153 ECDH_MODE_SOURCE_TEMPKEY . . . . .	742
20.71.2.154 ECDH_PREFIX_MODE . . . . .	742
20.71.2.155 ECDH_RSP_SIZE . . . . .	742
20.71.2.156 GENDIG_COUNT . . . . .	742
20.71.2.157 GENDIG_DATA_IDX . . . . .	742
20.71.2.158 GENDIG_KEYID_IDX . . . . .	742
20.71.2.159 GENDIG_RSP_SIZE . . . . .	743
20.71.2.160 GENDIG_ZONE_CONFIG . . . . .	743
20.71.2.161 GENDIG_ZONE_COUNTER . . . . .	743
20.71.2.162 GENDIG_ZONE_DATA . . . . .	743
20.71.2.163 GENDIG_ZONE_IDX . . . . .	743
20.71.2.164 GENDIG_ZONE_KEY_CONFIG . . . . .	743
20.71.2.165 GENDIG_ZONE_OTP . . . . .	744
20.71.2.166 GENDIG_ZONE_SHARED_NONCE . . . . .	744
20.71.2.167 GENKEY_COUNT . . . . .	744
20.71.2.168 GENKEY_COUNT_DATA . . . . .	744
20.71.2.169 GENKEY_DATA_IDX . . . . .	744
20.71.2.170 GENKEY_KEYID_IDX . . . . .	744
20.71.2.171 GENKEY_MODE_DIGEST . . . . .	745
20.71.2.172 GENKEY_MODE_IDX . . . . .	745
20.71.2.173 GENKEY_MODE_MASK . . . . .	745
20.71.2.174 GENKEY_MODE_PRIVATE . . . . .	745
20.71.2.175 GENKEY_MODE_PUBKEY_DIGEST . . . . .	745
20.71.2.176 GENKEY_MODE_PUBLIC . . . . .	745
20.71.2.177 GENKEY_OTHER_DATA_SIZE . . . . .	746
20.71.2.178 GENKEY_PRIVATE_TO_TEMPKEY . . . . .	746
20.71.2.179 GENKEY_RSP_SIZE_LONG . . . . .	746
20.71.2.180 GENKEY_RSP_SIZE_SHORT . . . . .	746
20.71.2.181 HMAC_COUNT . . . . .	746
20.71.2.182 HMAC_DIGEST_SIZE . . . . .	746
20.71.2.183 HMAC_KEYID_IDX . . . . .	747
20.71.2.184 HMAC_MODE_FLAG_FULLSN . . . . .	747
20.71.2.185 HMAC_MODE_FLAG_OTP64 . . . . .	747
20.71.2.186 HMAC_MODE_FLAG_OTP88 . . . . .	747
20.71.2.187 HMAC_MODE_FLAG_TK_NORAND . . . . .	747
20.71.2.188 HMAC_MODE_FLAG_TK_RAND . . . . .	747
20.71.2.189 HMAC_MODE_IDX . . . . .	748
20.71.2.190 HMAC_MODE_MASK . . . . .	748
20.71.2.191 HMAC_RSP_SIZE . . . . .	748
20.71.2.192 INFO_COUNT . . . . .	748
20.71.2.193 INFO_DRIVER_STATE_MASK . . . . .	748
20.71.2.194 INFO_MODE_GPIO . . . . .	748



20.71.2.195 INFO_MODE_KEY_VALID . . . . .	749
20.71.2.196 INFO_MODE_MAX . . . . .	749
20.71.2.197 INFO_MODE_REVISION . . . . .	749
20.71.2.198 INFO_MODE_STATE . . . . .	749
20.71.2.199 INFO_MODE_VOL_KEY_PERMIT . . . . .	749
20.71.2.200 INFO_NO_STATE . . . . .	749
20.71.2.201 INFO_OUTPUT_STATE_MASK . . . . .	750
20.71.2.202 INFO_PARAM1_IDX . . . . .	750
20.71.2.203 INFO_PARAM2_IDX . . . . .	750
20.71.2.204 INFO_PARAM2_LATCH_CLEAR . . . . .	750
20.71.2.205 INFO_PARAM2_LATCH_SET . . . . .	750
20.71.2.206 INFO_PARAM2_SET_LATCH_STATE . . . . .	750
20.71.2.207 INFO_RSP_SIZE . . . . .	751
20.71.2.208 INFO_SIZE . . . . .	751
20.71.2.209 KDF_DETAILS_AES_KEY_LOC_MASK . . . . .	751
20.71.2.210 KDF_DETAILS_HKDF_MSG_LOC_INPUT . . . . .	751
20.71.2.211 KDF_DETAILS_HKDF_MSG_LOC_IV . . . . .	751
20.71.2.212 KDF_DETAILS_HKDF_MSG_LOC_MASK . . . . .	751
20.71.2.213 KDF_DETAILS_HKDF_MSG_LOC_SLOT . . . . .	752
20.71.2.214 KDF_DETAILS_HKDF_MSG_LOC_TEMPKEY . . . . .	752
20.71.2.215 KDF_DETAILS_HKDF_ZERO_KEY . . . . .	752
20.71.2.216 KDF_DETAILS_IDX . . . . .	752
20.71.2.217 KDF_DETAILS_PRF_AEAD_MASK . . . . .	752
20.71.2.218 KDF_DETAILS_PRF_AEAD_MODE0 . . . . .	752
20.71.2.219 KDF_DETAILS_PRF_AEAD_MODE1 . . . . .	753
20.71.2.220 KDF_DETAILS_PRF_KEY_LEN_16 . . . . .	753
20.71.2.221 KDF_DETAILS_PRF_KEY_LEN_32 . . . . .	753
20.71.2.222 KDF_DETAILS_PRF_KEY_LEN_48 . . . . .	753
20.71.2.223 KDF_DETAILS_PRF_KEY_LEN_64 . . . . .	753
20.71.2.224 KDF_DETAILS_PRF_KEY_LEN_MASK . . . . .	753
20.71.2.225 KDF_DETAILS_PRF_TARGET_LEN_32 . . . . .	754
20.71.2.226 KDF_DETAILS_PRF_TARGET_LEN_64 . . . . .	754
20.71.2.227 KDF_DETAILS_PRF_TARGET_LEN_MASK . . . . .	754
20.71.2.228 KDF_DETAILS_SIZE . . . . .	754
20.71.2.229 KDF_KEYID_IDX . . . . .	754
20.71.2.230 KDF_MESSAGE_IDX . . . . .	754
20.71.2.231 KDF_MODE_ALG_AES . . . . .	755
20.71.2.232 KDF_MODE_ALG_HKDF . . . . .	755
20.71.2.233 KDF_MODE_ALG_MASK . . . . .	755
20.71.2.234 KDF_MODE_ALG_PRF . . . . .	755
20.71.2.235 KDF_MODE_IDX . . . . .	755
20.71.2.236 KDF_MODE_SOURCE_ALTKEYBUF . . . . .	755

20.71.2.237 KDF_MODE_SOURCE_MASK . . . . .	756
20.71.2.238 KDF_MODE_SOURCE_SLOT . . . . .	756
20.71.2.239 KDF_MODE_SOURCE_TEMPKEY . . . . .	756
20.71.2.240 KDF_MODE_SOURCE_TEMPKEY_UP . . . . .	756
20.71.2.241 KDF_MODE_TARGET_ALTKEYBUF . . . . .	756
20.71.2.242 KDF_MODE_TARGET_MASK . . . . .	756
20.71.2.243 KDF_MODE_TARGET_OUTPUT . . . . .	757
20.71.2.244 KDF_MODE_TARGET_OUTPUT_ENC . . . . .	757
20.71.2.245 KDF_MODE_TARGET_SLOT . . . . .	757
20.71.2.246 KDF_MODE_TARGET_TEMPKEY . . . . .	757
20.71.2.247 KDF_MODE_TARGET_TEMPKEY_UP . . . . .	757
20.71.2.248 LOCK_COUNT . . . . .	757
20.71.2.249 LOCK_RSP_SIZE . . . . .	758
20.71.2.250 LOCK_SUMMARY_IDX . . . . .	758
20.71.2.251 LOCK_ZONE_CONFIG . . . . .	758
20.71.2.252 LOCK_ZONE_DATA . . . . .	758
20.71.2.253 LOCK_ZONE_DATA_SLOT . . . . .	758
20.71.2.254 LOCK_ZONE_IDX . . . . .	758
20.71.2.255 LOCK_ZONE_MASK . . . . .	759
20.71.2.256 LOCK_ZONE_NO_CRC . . . . .	759
20.71.2.257 MAC_CHALLENGE_IDX . . . . .	759
20.71.2.258 MAC_CHALLENGE_SIZE . . . . .	759
20.71.2.259 MAC_COUNT_LONG . . . . .	759
20.71.2.260 MAC_COUNT_SHORT . . . . .	759
20.71.2.261 MAC_KEYID_IDX . . . . .	760
20.71.2.262 MAC_MODE_BLOCK1_TEMPKEY . . . . .	760
20.71.2.263 MAC_MODE_BLOCK2_TEMPKEY . . . . .	760
20.71.2.264 MAC_MODE_CHALLENGE . . . . .	760
20.71.2.265 MAC_MODE_IDX . . . . .	760
20.71.2.266 MAC_MODE_INCLUDE_OTP_64 . . . . .	760
20.71.2.267 MAC_MODE_INCLUDE_OTP_88 . . . . .	761
20.71.2.268 MAC_MODE_INCLUDE_SN . . . . .	761
20.71.2.269 MAC_MODE_MASK . . . . .	761
20.71.2.270 MAC_MODE_PASSTHROUGH . . . . .	761
20.71.2.271 MAC_MODE_PTNONCE_TEMPKEY . . . . .	761
20.71.2.272 MAC_MODE_SOURCE_FLAG_MATCH . . . . .	761
20.71.2.273 MAC_RSP_SIZE . . . . .	762
20.71.2.274 MAC_SIZE . . . . .	762
20.71.2.275 NONCE_COUNT_LONG . . . . .	762
20.71.2.276 NONCE_COUNT_LONG_64 . . . . .	762
20.71.2.277 NONCE_COUNT_SHORT . . . . .	762
20.71.2.278 NONCE_INPUT_IDX . . . . .	762

20.71.2.279 NONCE_MODE_IDX . . . . .	763
20.71.2.280 NONCE_MODE_INPUT_LEN_32 . . . . .	763
20.71.2.281 NONCE_MODE_INPUT_LEN_64 . . . . .	763
20.71.2.282 NONCE_MODE_INPUT_LEN_MASK . . . . .	763
20.71.2.283 NONCE_MODE_INVALID . . . . .	763
20.71.2.284 NONCE_MODE_MASK . . . . .	763
20.71.2.285 NONCE_MODE_NO_SEED_UPDATE . . . . .	764
20.71.2.286 NONCE_MODE_PASSTHROUGH . . . . .	764
20.71.2.287 NONCE_MODE_SEED_UPDATE . . . . .	764
20.71.2.288 NONCE_MODE_TARGET_ALTKEYBUF . . . . .	764
20.71.2.289 NONCE_MODE_TARGET_MASK . . . . .	764
20.71.2.290 NONCE_MODE_TARGET_MSGDIGBUF . . . . .	764
20.71.2.291 NONCE_MODE_TARGET_TEMPKEY . . . . .	765
20.71.2.292 NONCE_NUMIN_SIZE . . . . .	765
20.71.2.293 NONCE_NUMIN_SIZE_PASSTHROUGH . . . . .	765
20.71.2.294 NONCE_PARAM2_IDX . . . . .	765
20.71.2.295 NONCE_RSP_SIZE_LONG . . . . .	765
20.71.2.296 NONCE_RSP_SIZE_SHORT . . . . .	765
20.71.2.297 NONCE_ZERO_CALC_MASK . . . . .	766
20.71.2.298 NONCE_ZERO_CALC_RANDOM . . . . .	766
20.71.2.299 NONCE_ZERO_CALC_TEMPKEY . . . . .	766
20.71.2.300 OUTNONCE_SIZE . . . . .	766
20.71.2.301 PAUSE_COUNT . . . . .	766
20.71.2.302 PAUSE_PARAM2_IDX . . . . .	766
20.71.2.303 PAUSE_RSP_SIZE . . . . .	767
20.71.2.304 PAUSE_SELECT_IDX . . . . .	767
20.71.2.305 PRIVWRITE_COUNT . . . . .	767
20.71.2.306 PRIVWRITE_KEYID_IDX . . . . .	767
20.71.2.307 PRIVWRITE_MAC_IDX . . . . .	767
20.71.2.308 PRIVWRITE_MODE_ENCRYPT . . . . .	767
20.71.2.309 PRIVWRITE_RSP_SIZE . . . . .	768
20.71.2.310 PRIVWRITE_VALUE_IDX . . . . .	768
20.71.2.311 PRIVWRITE_ZONE_IDX . . . . .	768
20.71.2.312 PRIVWRITE_ZONE_MASK . . . . .	768
20.71.2.313 RANDOM_COUNT . . . . .	768
20.71.2.314 RANDOM_MODE_IDX . . . . .	768
20.71.2.315 RANDOM_NO_SEED_UPDATE . . . . .	769
20.71.2.316 RANDOM_NUM_SIZE . . . . .	769
20.71.2.317 RANDOM_PARAM2_IDX . . . . .	769
20.71.2.318 RANDOM_RSP_SIZE . . . . .	769
20.71.2.319 RANDOM_SEED_UPDATE . . . . .	769
20.71.2.320 READ_32_RSP_SIZE . . . . .	769

20.71.2.321 READ_4_RSP_SIZE . . . . .	770
20.71.2.322 READ_ADDR_IDX . . . . .	770
20.71.2.323 READ_COUNT . . . . .	770
20.71.2.324 READ_ZONE_IDX . . . . .	770
20.71.2.325 READ_ZONE_MASK . . . . .	770
20.71.2.326 RSA2048_KEY_SIZE . . . . .	770
20.71.2.327 SECUREBOOT_COUNT_DIG . . . . .	771
20.71.2.328 SECUREBOOT_COUNT_DIG_SIG . . . . .	771
20.71.2.329 SECUREBOOT_DIGEST_SIZE . . . . .	771
20.71.2.330 SECUREBOOT_MAC_SIZE . . . . .	771
20.71.2.331 SECUREBOOT_MODE_ENC_MAC_FLAG . . . . .	771
20.71.2.332 SECUREBOOT_MODE_FULL . . . . .	771
20.71.2.333 SECUREBOOT_MODE_FULL_COPY . . . . .	772
20.71.2.334 SECUREBOOT_MODE_FULL_STORE . . . . .	772
20.71.2.335 SECUREBOOT_MODE_IDX . . . . .	772
20.71.2.336 SECUREBOOT_MODE_MASK . . . . .	772
20.71.2.337 SECUREBOOT_MODE_PROHIBIT_FLAG . . . . .	772
20.71.2.338 SECUREBOOT_RSP_SIZE_MAC . . . . .	772
20.71.2.339 SECUREBOOT_RSP_SIZE_NO_MAC . . . . .	773
20.71.2.340 SECUREBOOT_SIGNATURE_SIZE . . . . .	773
20.71.2.341 SECUREBOOTCONFIG_MODE_DISABLED . . . . .	773
20.71.2.342 SECUREBOOTCONFIG_MODE_FULL_BOTH . . . . .	773
20.71.2.343 SECUREBOOTCONFIG_MODE_FULL_DIG . . . . .	773
20.71.2.344 SECUREBOOTCONFIG_MODE_FULL_SIG . . . . .	773
20.71.2.345 SECUREBOOTCONFIG_MODE_MASK . . . . .	774
20.71.2.346 SECUREBOOTCONFIG_OFFSET . . . . .	774
20.71.2.347 SELFTEST_COUNT . . . . .	774
20.71.2.348 SELFTEST_MODE_AES . . . . .	774
20.71.2.349 SELFTEST_MODE_ALL . . . . .	774
20.71.2.350 SELFTEST_MODE_ECDH . . . . .	774
20.71.2.351 SELFTEST_MODE_ECDSA_SIGN_VERIFY . . . . .	775
20.71.2.352 SELFTEST_MODE_IDX . . . . .	775
20.71.2.353 SELFTEST_MODE_RNG . . . . .	775
20.71.2.354 SELFTEST_MODE_SHA . . . . .	775
20.71.2.355 SELFTEST_RSP_SIZE . . . . .	775
20.71.2.356 SHA_COUNT_LONG . . . . .	775
20.71.2.357 SHA_COUNT_SHORT . . . . .	776
20.71.2.358 SHA_DATA_MAX . . . . .	776
20.71.2.359 SHA_MODE_608_HMAC_END . . . . .	776
20.71.2.360 SHA_MODE_HMAC_END . . . . .	776
20.71.2.361 SHA_MODE_HMAC_START . . . . .	776
20.71.2.362 SHA_MODE_HMAC_UPDATE . . . . .	776

20.71.2.363 SHA_MODE_MASK . . . . .	777
20.71.2.364 SHA_MODE_READ_CONTEXT . . . . .	777
20.71.2.365 SHA_MODE_SHA256_END . . . . .	777
20.71.2.366 SHA_MODE_SHA256_PUBLIC . . . . .	777
20.71.2.367 SHA_MODE_SHA256_START . . . . .	777
20.71.2.368 SHA_MODE_SHA256_UPDATE . . . . .	777
20.71.2.369 SHA_MODE_TARGET_MASK . . . . .	778
20.71.2.370 SHA_MODE_WRITE_CONTEXT . . . . .	778
20.71.2.371 SHA_RSP_SIZE . . . . .	778
20.71.2.372 SHA_RSP_SIZE_LONG . . . . .	778
20.71.2.373 SHA_RSP_SIZE_SHORT . . . . .	778
20.71.2.374 SIGN_COUNT . . . . .	778
20.71.2.375 SIGN_KEYID_IDX . . . . .	779
20.71.2.376 SIGN_MODE_EXTERNAL . . . . .	779
20.71.2.377 SIGN_MODE_IDX . . . . .	779
20.71.2.378 SIGN_MODE_INCLUDE_SN . . . . .	779
20.71.2.379 SIGN_MODE_INTERNAL . . . . .	779
20.71.2.380 SIGN_MODE_INVALIDATE . . . . .	779
20.71.2.381 SIGN_MODE_MASK . . . . .	780
20.71.2.382 SIGN_MODE_SOURCE_MASK . . . . .	780
20.71.2.383 SIGN_MODE_SOURCE_MSGDIGBUF . . . . .	780
20.71.2.384 SIGN_MODE_SOURCE_TEMPKEY . . . . .	780
20.71.2.385 SIGN_RSP_SIZE . . . . .	780
20.71.2.386 UPDATE_COUNT . . . . .	780
20.71.2.387 UPDATE_MODE_DEC_COUNTER . . . . .	781
20.71.2.388 UPDATE_MODE_IDX . . . . .	781
20.71.2.389 UPDATE_MODE_SELECTOR . . . . .	781
20.71.2.390 UPDATE_MODE_USER_EXTRA . . . . .	781
20.71.2.391 UPDATE_MODE_USER_EXTRA_ADD . . . . .	781
20.71.2.392 UPDATE_RSP_SIZE . . . . .	781
20.71.2.393 UPDATE_VALUE_IDX . . . . .	782
20.71.2.394 VERIFY_256_EXTERNAL_COUNT . . . . .	782
20.71.2.395 VERIFY_256_KEY_SIZE . . . . .	782
20.71.2.396 VERIFY_256_SIGNATURE_SIZE . . . . .	782
20.71.2.397 VERIFY_256_STORED_COUNT . . . . .	782
20.71.2.398 VERIFY_256_VALIDATE_COUNT . . . . .	782
20.71.2.399 VERIFY_283_EXTERNAL_COUNT . . . . .	783
20.71.2.400 VERIFY_283_KEY_SIZE . . . . .	783
20.71.2.401 VERIFY_283_SIGNATURE_SIZE . . . . .	783
20.71.2.402 VERIFY_283_STORED_COUNT . . . . .	783
20.71.2.403 VERIFY_283_VALIDATE_COUNT . . . . .	783
20.71.2.404 VERIFY_DATA_IDX . . . . .	783

20.71.2.405 VERIFY_KEY_B283 . . . . .	784
20.71.2.406 VERIFY_KEY_K283 . . . . .	784
20.71.2.407 VERIFY_KEY_P256 . . . . .	784
20.71.2.408 VERIFY_KEYID_IDX . . . . .	784
20.71.2.409 VERIFY_MODE_EXTERNAL . . . . .	784
20.71.2.410 VERIFY_MODE_IDX . . . . .	784
20.71.2.411 VERIFY_MODE_INVALIDATE . . . . .	785
20.71.2.412 VERIFY_MODE_MAC_FLAG . . . . .	785
20.71.2.413 VERIFY_MODE_MASK . . . . .	785
20.71.2.414 VERIFY_MODE_SOURCE_MASK . . . . .	785
20.71.2.415 VERIFY_MODE_SOURCE_MSGDIGBUF . . . . .	785
20.71.2.416 VERIFY_MODE_SOURCE_TEMPKEY . . . . .	785
20.71.2.417 VERIFY_MODE_STORED . . . . .	786
20.71.2.418 VERIFY_MODE_VALIDATE . . . . .	786
20.71.2.419 VERIFY_MODE_VALIDATE_EXTERNAL . . . . .	786
20.71.2.420 VERIFY_OTHER_DATA_SIZE . . . . .	786
20.71.2.421 VERIFY_RSP_SIZE . . . . .	786
20.71.2.422 VERIFY_RSP_SIZE_MAC . . . . .	786
20.71.2.423 WRITE_ADDR_IDX . . . . .	787
20.71.2.424 WRITE_MAC_SIZE . . . . .	787
20.71.2.425 WRITE_MAC_VL_IDX . . . . .	787
20.71.2.426 WRITE_MAC_VS_IDX . . . . .	787
20.71.2.427 WRITE_RSP_SIZE . . . . .	787
20.71.2.428 WRITE_VALUE_IDX . . . . .	787
20.71.2.429 WRITE_ZONE_DATA . . . . .	788
20.71.2.430 WRITE_ZONE_IDX . . . . .	788
20.71.2.431 WRITE_ZONE_MASK . . . . .	788
20.71.2.432 WRITE_ZONE_OTP . . . . .	788
20.71.2.433 WRITE_ZONE_WITH_MAC . . . . .	788
20.71.3 Function Documentation . . . . .	788
20.71.3.1 atAES() . . . . .	788
20.71.3.2 atCalcCrc() . . . . .	789
20.71.3.3 atCheckCrc() . . . . .	789
20.71.3.4 atCheckMAC() . . . . .	789
20.71.3.5 atCounter() . . . . .	790
20.71.3.6 atCRC() . . . . .	790
20.71.3.7 atDeriveKey() . . . . .	791
20.71.3.8 atECDH() . . . . .	791
20.71.3.9 atGenDig() . . . . .	791
20.71.3.10 atGenKey() . . . . .	792
20.71.3.11 atHMAC() . . . . .	792
20.71.3.12 atInfo() . . . . .	793

20.71.3.13 atlsECCFamily()	793
20.71.3.14 atlsSHAFamily()	793
20.71.3.15 atKDF()	794
20.71.3.16 atLock()	794
20.71.3.17 atMAC()	794
20.71.3.18 atNonce()	795
20.71.3.19 atPause()	795
20.71.3.20 atPrivWrite()	796
20.71.3.21 atRandom()	796
20.71.3.22 atRead()	796
20.71.3.23 atSecureBoot()	797
20.71.3.24 atSelfTest()	797
20.71.3.25 atSHA()	797
20.71.3.26 atSign()	799
20.71.3.27 atUpdateExtra()	799
20.71.3.28 atVerify()	800
20.71.3.29 atWrite()	800
20.71.3.30 isATCAError()	800
20.72 calib_counter.c File Reference	801
20.72.1 Detailed Description	801
20.73 calib_derivekey.c File Reference	801
20.73.1 Detailed Description	802
20.74 calib_ecdh.c File Reference	802
20.74.1 Detailed Description	803
20.74.2 Function Documentation	803
20.74.2.1 calib_ecdh_enc()	803
20.75 calib_execution.c File Reference	804
20.75.1 Detailed Description	804
20.75.2 Function Documentation	804
20.75.2.1 calib_execute_command()	804
20.76 calib_execution.h File Reference	805
20.76.1 Detailed Description	805
20.76.2 Macro Definition Documentation	805
20.76.2.1 ATCA_UNSUPPORTED_CMD	806
20.76.3 Function Documentation	806
20.76.3.1 calib_execute_command()	806
20.77 calib_gendig.c File Reference	806
20.77.1 Detailed Description	806
20.78 calib_genkey.c File Reference	807
20.78.1 Detailed Description	807
20.79 calib_hmac.c File Reference	807
20.79.1 Detailed Description	808

20.80 calib_info.c File Reference . . . . .	808
20.80.1 Detailed Description . . . . .	808
20.81 calib_kdf.c File Reference . . . . .	809
20.81.1 Detailed Description . . . . .	809
20.82 calib_lock.c File Reference . . . . .	809
20.82.1 Detailed Description . . . . .	810
20.83 calib_mac.c File Reference . . . . .	810
20.83.1 Detailed Description . . . . .	810
20.84 calib_nonce.c File Reference . . . . .	811
20.84.1 Detailed Description . . . . .	811
20.85 calib_privwrite.c File Reference . . . . .	811
20.85.1 Detailed Description . . . . .	812
20.85.2 Function Documentation . . . . .	812
20.85.2.1 calib_priv_write() . . . . .	812
20.86 calib_random.c File Reference . . . . .	813
20.86.1 Detailed Description . . . . .	813
20.87 calib_read.c File Reference . . . . .	813
20.87.1 Detailed Description . . . . .	814
20.87.2 Function Documentation . . . . .	814
20.87.2.1 calib_read_enc() . . . . .	815
20.88 calib_secureboot.c File Reference . . . . .	815
20.88.1 Detailed Description . . . . .	816
20.89 calib_selftest.c File Reference . . . . .	816
20.89.1 Detailed Description . . . . .	816
20.90 calib_sha.c File Reference . . . . .	816
20.90.1 Detailed Description . . . . .	818
20.91 calib_sign.c File Reference . . . . .	818
20.91.1 Detailed Description . . . . .	818
20.92 calib_updateextra.c File Reference . . . . .	819
20.92.1 Detailed Description . . . . .	819
20.93 calib_verify.c File Reference . . . . .	819
20.93.1 Detailed Description . . . . .	820
20.94 calib_write.c File Reference . . . . .	820
20.94.1 Detailed Description . . . . .	821
20.94.2 Function Documentation . . . . .	821
20.94.2.1 calib_write_enc() . . . . .	822
20.95 cryptoauthlib.h File Reference . . . . .	822
20.95.1 Detailed Description . . . . .	823
20.95.2 Macro Definition Documentation . . . . .	823
20.95.2.1 ATCA_AES128_BLOCK_SIZE . . . . .	823
20.95.2.2 ATCA_AES128_KEY_SIZE . . . . .	824
20.95.2.3 ATCA_ATECC608_SUPPORT . . . . .	824



20.95.2.4 ATCA_CA_SUPPORT . . . . .	824
20.95.2.5 ATCA_DLL . . . . .	824
20.95.2.6 ATCA_ECC_SUPPORT . . . . .	824
20.95.2.7 ATCA_ECCP256_KEY_SIZE . . . . .	824
20.95.2.8 ATCA_ECCP256_PUBKEY_SIZE . . . . .	824
20.95.2.9 ATCA_ECCP256_SIG_SIZE . . . . .	824
20.95.2.10 ATCA_SHA256_BLOCK_SIZE . . . . .	825
20.95.2.11 ATCA_SHA256_DIGEST_SIZE . . . . .	825
20.95.2.12 ATCA_SHA_SUPPORT . . . . .	825
20.95.2.13 ATCA_STRINGIFY . . . . .	825
20.95.2.14 ATCA_TA_SUPPORT . . . . .	825
20.95.2.15 ATCA_TOSTRING . . . . .	825
20.95.2.16 ATCA_TRACE . . . . .	825
20.95.2.17 ATCA_ZONE_CONFIG . . . . .	826
20.95.2.18 ATCA_ZONE_DATA . . . . .	826
20.95.2.19 ATCA_ZONE_OTP . . . . .	826
20.95.2.20 SHA_MODE_TARGET_MSGDIGBUF . . . . .	826
20.95.2.21 SHA_MODE_TARGET_OUT_ONLY . . . . .	826
20.95.2.22 SHA_MODE_TARGET_TEMPKEY . . . . .	826
20.96 cryptoki.h File Reference . . . . .	826
20.96.1 Macro Definition Documentation . . . . .	827
20.96.1.1 CK_CALLBACK_FUNCTION . . . . .	827
20.96.1.2 CK_DECLARE_FUNCTION . . . . .	827
20.96.1.3 CK_DECLARE_FUNCTION_POINTER . . . . .	827
20.96.1.4 CK_PTR . . . . .	827
20.96.1.5 NULL_PTR . . . . .	828
20.96.1.6 PKCS11_API . . . . .	828
20.96.1.7 PKCS11_HELPER_DLL_EXPORT . . . . .	828
20.96.1.8 PKCS11_HELPER_DLL_IMPORT . . . . .	828
20.96.1.9 PKCS11_HELPER_DLL_LOCAL . . . . .	828
20.96.1.10 PKCS11_LOCAL . . . . .	828
20.97 example_cert_chain.c File Reference . . . . .	828
20.97.1 Variable Documentation . . . . .	829
20.97.1.1 g_cert_def_0_root . . . . .	829
20.97.1.2 g_cert_def_1_signer . . . . .	829
20.97.1.3 g_cert_def_2_device . . . . .	829
20.97.1.4 g_cert_elements_1_signer . . . . .	829
20.97.1.5 g_cert_template_1_signer . . . . .	829
20.97.1.6 g_cert_template_2_device . . . . .	830
20.98 example_cert_chain.h File Reference . . . . .	830
20.98.1 Variable Documentation . . . . .	830
20.98.1.1 g_cert_def_1_signer . . . . .	830

20.98.1.2 g_cert_def_2_device . . . . .	830
20.99 example_pkcs11_config.c File Reference . . . . .	831
20.99.1 Macro Definition Documentation . . . . .	831
20.99.1.1 pkcs11configLABEL_DEVICE_CERTIFICATE_FOR_TLS . . . . .	831
20.99.1.2 pkcs11configLABEL_DEVICE_PRIVATE_KEY_FOR_TLS . . . . .	831
20.99.1.3 pkcs11configLABEL_DEVICE_PUBLIC_KEY_FOR_TLS . . . . .	832
20.99.1.4 pkcs11configLABEL_JITP_CERTIFICATE . . . . .	832
20.99.2 Function Documentation . . . . .	832
20.99.2.1 pkcs11_config_cert() . . . . .	832
20.99.2.2 pkcs11_config_key() . . . . .	832
20.99.2.3 pkcs11_config_load_objects() . . . . .	832
20.99.3 Variable Documentation . . . . .	832
20.99.3.1 atecc608_config . . . . .	833
20.100 hal_all_platforms_kit_hidapi.c File Reference . . . . .	833
20.100.1 Detailed Description . . . . .	834
20.101 hal_all_platforms_kit_hidapi.h File Reference . . . . .	834
20.101.1 Detailed Description . . . . .	834
20.102 hal_esp32_i2c.c File Reference . . . . .	835
20.102.1 Macro Definition Documentation . . . . .	836
20.102.1.1 ACK_CHECK_DIS . . . . .	836
20.102.1.2 ACK_CHECK_EN . . . . .	836
20.102.1.3 ACK_VAL . . . . .	836
20.102.1.4 LOG_LOCAL_LEVEL . . . . .	836
20.102.1.5 MAX_I2C_BUSES . . . . .	837
20.102.1.6 NACK_VAL . . . . .	837
20.102.1.7 SCL_PIN . . . . .	837
20.102.1.8 SDA_PIN . . . . .	837
20.102.2 Typedef Documentation . . . . .	837
20.102.2.1 ATCAI2CMaster_t . . . . .	837
20.102.3 Function Documentation . . . . .	837
20.102.3.1 hal_i2c_change_baud() . . . . .	837
20.102.3.2 hal_i2c_discover_buses() . . . . .	838
20.102.3.3 hal_i2c_discover_devices() . . . . .	839
20.102.3.4 hal_i2c_idle() . . . . .	839
20.102.3.5 hal_i2c_init() . . . . .	840
20.102.3.6 hal_i2c_post_init() . . . . .	842
20.102.3.7 hal_i2c_receive() . . . . .	843
20.102.3.8 hal_i2c_release() . . . . .	844
20.102.3.9 hal_i2c_send() . . . . .	845
20.102.3.10 hal_i2c_sleep() . . . . .	846
20.102.3.11 hal_i2c_wake() . . . . .	846
20.102.4 Variable Documentation . . . . .	846

20.102.4.1 conf . . . . .	846
20.102.4.2 i2c_bus_ref_ct . . . . .	847
20.102.4.3 i2c_hal_data . . . . .	847
20.102.4.4 TAG . . . . .	847
20.103 hal_esp32_timer.c File Reference . . . . .	847
20.103.1 Function Documentation . . . . .	847
20.103.1.1 atca_delay_ms() . . . . .	847
20.103.1.2 ets_delay_us() . . . . .	847
20.104 hal_freertos.c File Reference . . . . .	848
20.104.1 Detailed Description . . . . .	848
20.104.2 Macro Definition Documentation . . . . .	848
20.104.2.1 ATCA_MUTEX_TIMEOUT . . . . .	848
20.105 hal_i2c_harmony.c File Reference . . . . .	848
20.105.1 Detailed Description . . . . .	849
20.106 hal_i2c_start.c File Reference . . . . .	849
20.106.1 Detailed Description . . . . .	850
20.107 hal_i2c_start.h File Reference . . . . .	850
20.107.1 Detailed Description . . . . .	851
20.108 hal_linux.c File Reference . . . . .	851
20.108.1 Detailed Description . . . . .	852
20.109 hal_linux_i2c_userspace.c File Reference . . . . .	852
20.109.1 Detailed Description . . . . .	853
20.110 hal_linux_i2c_userspace.h File Reference . . . . .	853
20.110.1 Detailed Description . . . . .	853
20.111 hal_linux_kit_hid.c File Reference . . . . .	853
20.111.1 Detailed Description . . . . .	854
20.112 hal_linux_kit_hid.h File Reference . . . . .	854
20.112.1 Detailed Description . . . . .	855
20.113 hal_linux_spi_userspace.c File Reference . . . . .	855
20.113.1 Function Documentation . . . . .	856
20.113.1.1 hal_spi_discover_buses() . . . . .	856
20.113.1.2 hal_spi_discover_devices() . . . . .	856
20.113.1.3 hal_spi_idle() . . . . .	856
20.113.1.4 hal_spi_init() . . . . .	857
20.113.1.5 hal_spi_open_file() . . . . .	857
20.113.1.6 hal_spi_post_init() . . . . .	857
20.113.1.7 hal_spi_receive() . . . . .	858
20.113.1.8 hal_spi_release() . . . . .	858
20.113.1.9 hal_spi_send() . . . . .	858
20.113.1.10 hal_spi_sleep() . . . . .	859
20.113.1.11 hal_spi_wake() . . . . .	859
20.114 hal_linux_spi_userspace.h File Reference . . . . .	860

20.114.1 Detailed Description . . . . .	860
20.115 hal_sam0_i2c_asf.c File Reference . . . . .	860
20.115.1 Detailed Description . . . . .	861
20.116 hal_sam0_i2c_asf.h File Reference . . . . .	861
20.116.1 Detailed Description . . . . .	862
20.116.2 Typedef Documentation . . . . .	862
20.116.2.1 i2c_sam0_instance_t . . . . .	862
20.116.2.2 sam0_change_baudrate . . . . .	862
20.117 hal_sam_i2c_asf.c File Reference . . . . .	862
20.117.1 Detailed Description . . . . .	863
20.118 hal_sam_i2c_asf.h File Reference . . . . .	863
20.118.1 Detailed Description . . . . .	864
20.119 hal_sam_timer_asf.c File Reference . . . . .	864
20.119.1 Detailed Description . . . . .	864
20.120 hal_spi_harmony.c File Reference . . . . .	865
20.120.1 Detailed Description . . . . .	865
20.121 hal_swi_uart.c File Reference . . . . .	866
20.121.1 Detailed Description . . . . .	866
20.122 hal_swi_uart.h File Reference . . . . .	867
20.122.1 Detailed Description . . . . .	867
20.123 hal_timer_start.c File Reference . . . . .	867
20.123.1 Detailed Description . . . . .	868
20.124 hal_uc3_i2c_asf.c File Reference . . . . .	868
20.124.1 Detailed Description . . . . .	869
20.125 hal_uc3_i2c_asf.h File Reference . . . . .	869
20.125.1 Detailed Description . . . . .	869
20.126 hal_uc3_timer_asf.c File Reference . . . . .	870
20.126.1 Detailed Description . . . . .	870
20.127 hal_win_kit_hid.c File Reference . . . . .	870
20.127.1 Detailed Description . . . . .	871
20.128 hal_win_kit_hid.h File Reference . . . . .	871
20.128.1 Detailed Description . . . . .	872
20.129 hal_windows.c File Reference . . . . .	872
20.129.1 Detailed Description . . . . .	873
20.130 io_protection_key.h File Reference . . . . .	873
20.130.1 Detailed Description . . . . .	873
20.130.2 Function Documentation . . . . .	873
20.130.2.1 io_protection_get_key() . . . . .	873
20.130.2.2 io_protection_set_key() . . . . .	873
20.131 kit_phy.h File Reference . . . . .	874
20.131.1 Detailed Description . . . . .	874
20.132 kit_protocol.c File Reference . . . . .	874

20.132.1 Detailed Description . . . . .	875
20.133 kit_protocol.h File Reference . . . . .	875
20.133.1 Detailed Description . . . . .	876
20.134 license.txt File Reference . . . . .	876
20.134.1 Function Documentation . . . . .	882
20.134.1.1 DAMAGES() . . . . .	882
20.134.1.2 or() . . . . .	883
20.134.1.3 software() . . . . .	883
20.134.1.4 TORT() . . . . .	883
20.134.2 Variable Documentation . . . . .	884
20.134.2.1 ANY . . . . .	884
20.134.2.2 CAUSED . . . . .	884
20.134.2.3 CONTRACT . . . . .	884
20.134.2.4 DAMAGE . . . . .	885
20.134.2.5 DIRECT . . . . .	885
20.134.2.6 EXEMPLARY . . . . .	886
20.134.2.7 EXPRESS . . . . .	886
20.134.2.8 FEES . . . . .	886
20.134.2.9 forms . . . . .	887
20.134.2.10 Foundation . . . . .	887
20.134.2.11 INCIDENTAL . . . . .	887
20.134.2.12 INCLUDING . . . . .	888
20.134.2.13 INDIRECT . . . . .	888
20.134.2.14 INFRINGEMENT . . . . .	888
20.134.2.15 LAW . . . . .	889
20.134.2.16 LIABILITY . . . . .	889
20.134.2.17 license . . . . .	889
20.134.2.18 License . . . . .	890
20.134.2.19 LOSS . . . . .	890
20.134.2.20 MERCHANTABILITY . . . . .	890
20.134.2.21 met . . . . .	890
20.134.2.22 modification . . . . .	891
20.134.2.23 not . . . . .	891
20.134.2.24 notice . . . . .	891
20.134.2.25 Ott . . . . .	891
20.134.2.26 PUNITIVE . . . . .	892
20.134.2.27 SOFTWARE . . . . .	892
20.134.2.28 SPECIAL . . . . .	892
20.134.2.29 STATUTORY . . . . .	893
20.134.2.30 systemd . . . . .	893
20.134.2.31 terms . . . . .	893
20.134.2.32 TO . . . . .	893

20.134.2.33 WARRANTIES . . . . .	894
20.134.2.34 WARRANTY . . . . .	894
20.135 pkcs11.h File Reference . . . . .	894
20.135.1 Macro Definition Documentation . . . . .	894
20.135.1.1 __PASTE . . . . .	895
20.135.1.2 CK_NEED_ARG_LIST [1/2] . . . . .	895
20.135.1.3 CK_NEED_ARG_LIST [2/2] . . . . .	895
20.135.1.4 CK_PKCS11_FUNCTION_INFO [1/3] . . . . .	895
20.135.1.5 CK_PKCS11_FUNCTION_INFO [2/3] . . . . .	895
20.135.1.6 CK_PKCS11_FUNCTION_INFO [3/3] . . . . .	895
20.136 pkcs11_attrib.c File Reference . . . . .	895
20.136.1 Detailed Description . . . . .	896
20.137 pkcs11_attrib.h File Reference . . . . .	896
20.137.1 Detailed Description . . . . .	897
20.137.2 Typedef Documentation . . . . .	897
20.137.2.1 attrib_f . . . . .	897
20.137.2.2 pkcs11_attrib_model . . . . .	897
20.137.2.3 pkcs11_attrib_model_ptr . . . . .	897
20.138 pkcs11_cert.c File Reference . . . . .	897
20.138.1 Detailed Description . . . . .	898
20.139 pkcs11_cert.h File Reference . . . . .	898
20.139.1 Detailed Description . . . . .	899
20.140 pkcs11_config.c File Reference . . . . .	899
20.140.1 Detailed Description . . . . .	899
20.141 pkcs11_debug.c File Reference . . . . .	900
20.141.1 Detailed Description . . . . .	900
20.142 pkcs11_debug.h File Reference . . . . .	900
20.142.1 Detailed Description . . . . .	900
20.142.2 Macro Definition Documentation . . . . .	900
20.142.2.1 PKCS11_DEBUG . . . . .	901
20.142.2.2 pkcs11_debug_attributes . . . . .	901
20.142.2.3 PKCS11_DEBUG_NOFILE . . . . .	901
20.142.2.4 PKCS11_DEBUG_RETURN . . . . .	901
20.143 pkcs11_digest.c File Reference . . . . .	901
20.143.1 Function Documentation . . . . .	902
20.143.1.1 pkcs11_digest() . . . . .	902
20.143.1.2 pkcs11_digest_final() . . . . .	902
20.143.1.3 pkcs11_digest_init() . . . . .	902
20.143.1.4 pkcs11_digest_update() . . . . .	902
20.144 pkcs11_digest.h File Reference . . . . .	903
20.144.1 Detailed Description . . . . .	903
20.144.2 Function Documentation . . . . .	903

20.144.2.1 pkcs11_digest()	903
20.144.2.2 pkcs11_digest_final()	904
20.144.2.3 pkcs11_digest_init()	904
20.144.2.4 pkcs11_digest_update()	904
20.145 pkcs11_find.c File Reference	904
20.145.1 Detailed Description	905
20.146 pkcs11_find.h File Reference	905
20.146.1 Detailed Description	905
20.147 pkcs11_info.c File Reference	905
20.147.1 Detailed Description	906
20.148 pkcs11_info.h File Reference	906
20.148.1 Detailed Description	906
20.149 pkcs11_init.c File Reference	907
20.149.1 Detailed Description	907
20.150 pkcs11_init.h File Reference	907
20.150.1 Detailed Description	908
20.150.2 Typedef Documentation	908
20.150.2.1 pkcs11_lib_ctx	908
20.150.2.2 pkcs11_lib_ctx_ptr	908
20.151 pkcs11_key.c File Reference	909
20.151.1 Detailed Description	909
20.152 pkcs11_key.h File Reference	910
20.152.1 Detailed Description	910
20.153 pkcs11_main.c File Reference	910
20.153.1 Detailed Description	914
20.154 pkcs11_mech.c File Reference	914
20.154.1 Detailed Description	915
20.155 pkcs11_mech.h File Reference	915
20.155.1 Detailed Description	916
20.156 pkcs11_object.c File Reference	916
20.156.1 Detailed Description	917
20.157 pkcs11_object.h File Reference	917
20.157.1 Detailed Description	918
20.157.2 Macro Definition Documentation	918
20.157.2.1 PKCS11_OBJECT_FLAG_DESTROYABLE	918
20.157.2.2 PKCS11_OBJECT_FLAG_DYNAMIC	919
20.157.2.3 PKCS11_OBJECT_FLAG_MODIFIABLE	919
20.157.2.4 PKCS11_OBJECT_FLAG_SENSITIVE	919
20.157.2.5 PKCS11_OBJECT_FLAG_TA_TYPE	919
20.157.2.6 PKCS11_OBJECT_FLAG_TRUST_TYPE	919
20.157.3 Typedef Documentation	919
20.157.3.1 pkcs11_object	919

20.157.3.2 pkcs11_object_cache_t . . . . .	919
20.157.3.3 pkcs11_object_ptr . . . . .	920
20.158 pkcs11_os.c File Reference . . . . .	920
20.158.1 Detailed Description . . . . .	920
20.159 pkcs11_os.h File Reference . . . . .	920
20.159.1 Detailed Description . . . . .	921
20.159.2 Macro Definition Documentation . . . . .	921
20.159.2.1 pkcs11_os_free . . . . .	921
20.159.2.2 pkcs11_os_malloc . . . . .	921
20.160 pkcs11_session.c File Reference . . . . .	921
20.160.1 Detailed Description . . . . .	922
20.161 pkcs11_session.h File Reference . . . . .	922
20.161.1 Detailed Description . . . . .	922
20.161.2 Typedef Documentation . . . . .	923
20.161.2.1 pkcs11_session_ctx . . . . .	923
20.161.2.2 pkcs11_session_ctx_ptr . . . . .	923
20.161.3 Function Documentation . . . . .	923
20.161.3.1 pkcs11_session_authorize() . . . . .	923
20.162 pkcs11_signature.c File Reference . . . . .	923
20.162.1 Detailed Description . . . . .	924
20.163 pkcs11_signature.h File Reference . . . . .	924
20.163.1 Detailed Description . . . . .	925
20.164 pkcs11_slot.c File Reference . . . . .	925
20.164.1 Detailed Description . . . . .	926
20.165 pkcs11_slot.h File Reference . . . . .	926
20.165.1 Detailed Description . . . . .	927
20.165.2 Typedef Documentation . . . . .	927
20.165.2.1 pkcs11_slot_ctx . . . . .	927
20.165.2.2 pkcs11_slot_ctx_ptr . . . . .	927
20.166 pkcs11_token.c File Reference . . . . .	927
20.166.1 Detailed Description . . . . .	928
20.167 pkcs11_token.h File Reference . . . . .	928
20.167.1 Detailed Description . . . . .	929
20.168 pkcs11_util.c File Reference . . . . .	929
20.168.1 Detailed Description . . . . .	929
20.169 pkcs11_util.h File Reference . . . . .	929
20.169.1 Detailed Description . . . . .	930
20.169.2 Macro Definition Documentation . . . . .	930
20.169.2.1 PKCS11_UTIL_ARRAY_SIZE . . . . .	930
20.170 pkcs11f.h File Reference . . . . .	930
20.171 pkcs11t.h File Reference . . . . .	930
20.171.1 Macro Definition Documentation . . . . .	948



---

20.171.1.1 CK_CERTIFICATE_CATEGORY_AUTHORITY . . . . .	948
20.171.1.2 CK_CERTIFICATE_CATEGORY_OTHER_ENTITY . . . . .	948
20.171.1.3 CK_CERTIFICATE_CATEGORY_TOKEN_USER . . . . .	949
20.171.1.4 CK_CERTIFICATE_CATEGORY_UNSPECIFIED . . . . .	949
20.171.1.5 CK_EFFECTIVELY_INFINITE . . . . .	949
20.171.1.6 CK_FALSE . . . . .	949
20.171.1.7 CK_INVALID_HANDLE . . . . .	949
20.171.1.8 CK_OTP_CHALLENGE . . . . .	949
20.171.1.9 CK_OTP_COUNTER . . . . .	949
20.171.1.10 CK_OTP_FLAGS . . . . .	949
20.171.1.11 CK_OTP_FORMAT_ALPHANUMERIC . . . . .	950
20.171.1.12 CK_OTP_FORMAT_BINARY . . . . .	950
20.171.1.13 CK_OTP_FORMAT_DECIMAL . . . . .	950
20.171.1.14 CK_OTP_FORMAT_HEXADECIMAL . . . . .	950
20.171.1.15 CK_OTP_OUTPUT_FORMAT . . . . .	950
20.171.1.16 CK_OTP_OUTPUT_LENGTH . . . . .	950
20.171.1.17 CK_OTP_PARAM_IGNORED . . . . .	950
20.171.1.18 CK_OTP_PARAM_MANDATORY . . . . .	950
20.171.1.19 CK_OTP_PARAM_OPTIONAL . . . . .	951
20.171.1.20 CK_OTP_PIN . . . . .	951
20.171.1.21 CK_OTP_TIME . . . . .	951
20.171.1.22 CK_OTP_VALUE . . . . .	951
20.171.1.23 CK_SECURITY_DOMAIN_MANUFACTURER . . . . .	951
20.171.1.24 CK_SECURITY_DOMAIN_OPERATOR . . . . .	951
20.171.1.25 CK_SECURITY_DOMAIN_THIRD_PARTY . . . . .	951
20.171.1.26 CK_SECURITY_DOMAIN_UNSPECIFIED . . . . .	951
20.171.1.27 CK_TRUE . . . . .	952
20.171.1.28 CK_UNAVAILABLE_INFORMATION . . . . .	952
20.171.1.29 CKA_AC_ISSUER . . . . .	952
20.171.1.30 CKA_ALLOWED_MECHANISMS . . . . .	952
20.171.1.31 CKA_ALWAYS_AUTHENTICATE . . . . .	952
20.171.1.32 CKA_ALWAYS_SENSITIVE . . . . .	952
20.171.1.33 CKA_APPLICATION . . . . .	952
20.171.1.34 CKA_ATTR_TYPES . . . . .	952
20.171.1.35 CKA_AUTH_PIN_FLAGS . . . . .	953
20.171.1.36 CKA_BASE . . . . .	953
20.171.1.37 CKA_BITS_PER_PIXEL . . . . .	953
20.171.1.38 CKA_CERTIFICATE_CATEGORY . . . . .	953
20.171.1.39 CKA_CERTIFICATE_TYPE . . . . .	953
20.171.1.40 CKA_CHAR_COLUMNS . . . . .	953
20.171.1.41 CKA_CHAR_ROWS . . . . .	953
20.171.1.42 CKA_CHAR_SETS . . . . .	953

20.171.1.43 CKA_CHECK_VALUE . . . . .	954
20.171.1.44 CKA_CLASS . . . . .	954
20.171.1.45 CKA_COEFFICIENT . . . . .	954
20.171.1.46 CKA_COLOR . . . . .	954
20.171.1.47 CKA_COPYABLE . . . . .	954
20.171.1.48 CKA_DECRYPT . . . . .	954
20.171.1.49 CKA_DEFAULT_CMS_ATTRIBUTES . . . . .	954
20.171.1.50 CKA_DERIVE . . . . .	954
20.171.1.51 CKA_DERIVE_TEMPLATE . . . . .	955
20.171.1.52 CKA_DESTROYABLE . . . . .	955
20.171.1.53 CKA_EC_PARAMS . . . . .	955
20.171.1.54 CKA_EC_POINT . . . . .	955
20.171.1.55 CKA_ECDSA_PARAMS . . . . .	955
20.171.1.56 CKA_ENCODING_METHODS . . . . .	955
20.171.1.57 CKA_ENCRYPT . . . . .	955
20.171.1.58 CKA_END_DATE . . . . .	955
20.171.1.59 CKA_EXPONENT_1 . . . . .	956
20.171.1.60 CKA_EXPONENT_2 . . . . .	956
20.171.1.61 CKA_EXTRACTABLE . . . . .	956
20.171.1.62 CKA_GOST28147_PARAMS . . . . .	956
20.171.1.63 CKA_GOSTR3410_PARAMS . . . . .	956
20.171.1.64 CKA_GOSTR3411_PARAMS . . . . .	956
20.171.1.65 CKA_HAS_RESET . . . . .	956
20.171.1.66 CKA_HASH_OF_ISSUER_PUBLIC_KEY . . . . .	956
20.171.1.67 CKA_HASH_OF_SUBJECT_PUBLIC_KEY . . . . .	957
20.171.1.68 CKA_HW_FEATURE_TYPE . . . . .	957
20.171.1.69 CKA_ID . . . . .	957
20.171.1.70 CKA_ISSUER . . . . .	957
20.171.1.71 CKA_JAVA_MIDP_SECURITY_DOMAIN . . . . .	957
20.171.1.72 CKA_KEY_GEN_MECHANISM . . . . .	957
20.171.1.73 CKA_KEY_TYPE . . . . .	957
20.171.1.74 CKA_LABEL . . . . .	957
20.171.1.75 CKA_LOCAL . . . . .	958
20.171.1.76 CKA_MECHANISM_TYPE . . . . .	958
20.171.1.77 CKA_MIME_TYPES . . . . .	958
20.171.1.78 CKA_MODIFIABLE . . . . .	958
20.171.1.79 CKA_MODULUS . . . . .	958
20.171.1.80 CKA_MODULUS_BITS . . . . .	958
20.171.1.81 CKA_NAME_HASH_ALGORITHM . . . . .	958
20.171.1.82 CKA_NEVER_EXTRACTABLE . . . . .	958
20.171.1.83 CKA_OBJECT_ID . . . . .	959
20.171.1.84 CKA_OTP_CHALLENGE_REQUIREMENT . . . . .	959

20.171.1.85 CKA_OTP_COUNTER . . . . .	959
20.171.1.86 CKA_OTP_COUNTER_REQUIREMENT . . . . .	959
20.171.1.87 CKA_OTP_FORMAT . . . . .	959
20.171.1.88 CKA_OTP_LENGTH . . . . .	959
20.171.1.89 CKA_OTP_PIN_REQUIREMENT . . . . .	959
20.171.1.90 CKA_OTP_SERVICE_IDENTIFIER . . . . .	959
20.171.1.91 CKA_OTP_SERVICE_LOGO . . . . .	960
20.171.1.92 CKA_OTP_SERVICE_LOGO_TYPE . . . . .	960
20.171.1.93 CKA_OTP_TIME . . . . .	960
20.171.1.94 CKA_OTP_TIME_INTERVAL . . . . .	960
20.171.1.95 CKA_OTP_TIME_REQUIREMENT . . . . .	960
20.171.1.96 CKA_OTP_USER_FRIENDLY_MODE . . . . .	960
20.171.1.97 CKA_OTP_USER_IDENTIFIER . . . . .	960
20.171.1.98 CKA_OWNER . . . . .	960
20.171.1.99 CKA_PIXEL_X . . . . .	961
20.171.1.100 CKA_PIXEL_Y . . . . .	961
20.171.1.101 CKA_PRIME . . . . .	961
20.171.1.102 CKA_PRIME_1 . . . . .	961
20.171.1.103 CKA_PRIME_2 . . . . .	961
20.171.1.104 CKA_PRIME_BITS . . . . .	961
20.171.1.105 CKA_PRIVATE . . . . .	961
20.171.1.106 CKA_PRIVATE_EXPONENT . . . . .	961
20.171.1.107 CKA_PUBLIC_EXPONENT . . . . .	962
20.171.1.108 CKA_PUBLIC_KEY_INFO . . . . .	962
20.171.1.109 CKA_REQUIRED_CMS_ATTRIBUTES . . . . .	962
20.171.1.110 CKA_RESET_ON_INIT . . . . .	962
20.171.1.111 CKA_RESOLUTION . . . . .	962
20.171.1.112 CKA_SECONDARY_AUTH . . . . .	962
20.171.1.113 CKA_SENSITIVE . . . . .	962
20.171.1.114 CKA_SERIAL_NUMBER . . . . .	962
20.171.1.115 CKA_SIGN . . . . .	963
20.171.1.116 CKA_SIGN_RECOVER . . . . .	963
20.171.1.117 CKA_START_DATE . . . . .	963
20.171.1.118 CKA_SUB_PRIME_BITS . . . . .	963
20.171.1.119 CKA_SUBJECT . . . . .	963
20.171.1.120 CKA_SUBPRIME . . . . .	963
20.171.1.121 CKA_SUBPRIME_BITS . . . . .	963
20.171.1.122 CKA_SUPPORTED_CMS_ATTRIBUTES . . . . .	963
20.171.1.123 CKA_TOKEN . . . . .	964
20.171.1.124 CKA_TRUSTED . . . . .	964
20.171.1.125 CKA_UNWRAP . . . . .	964
20.171.1.126 CKA_UNWRAP_TEMPLATE . . . . .	964

20.171.1.127 CKA_URL . . . . .	964
20.171.1.128 CKA_VALUE . . . . .	964
20.171.1.129 CKA_VALUE_BITS . . . . .	964
20.171.1.130 CKA_VALUE_LEN . . . . .	964
20.171.1.131 CKA_VENDOR_DEFINED . . . . .	965
20.171.1.132 CKA_VERIFY . . . . .	965
20.171.1.133 CKA_VERIFY_RECOVER . . . . .	965
20.171.1.134 CKA_WRAP . . . . .	965
20.171.1.135 CKA_WRAP_TEMPLATE . . . . .	965
20.171.1.136 CKA_WRAP_WITH_TRUSTED . . . . .	965
20.171.1.137 CKC_OPENPGP . . . . .	965
20.171.1.138 CKC_VENDOR_DEFINED . . . . .	965
20.171.1.139 CKC_WTLS . . . . .	966
20.171.1.140 CKC_X_509 . . . . .	966
20.171.1.141 CKC_X_509_ATTR_CERT . . . . .	966
20.171.1.142 CKD_CPDIVERSIFY_KDF . . . . .	966
20.171.1.143 CKD_NULL . . . . .	966
20.171.1.144 CKD_SHA1_KDF . . . . .	966
20.171.1.145 CKD_SHA1_KDF_ASN1 . . . . .	966
20.171.1.146 CKD_SHA1_KDF_CONCATENATE . . . . .	966
20.171.1.147 CKD_SHA224_KDF . . . . .	967
20.171.1.148 CKD_SHA256_KDF . . . . .	967
20.171.1.149 CKD_SHA384_KDF . . . . .	967
20.171.1.150 CKD_SHA512_KDF . . . . .	967
20.171.1.151 CKF_ARRAY_ATTRIBUTE . . . . .	967
20.171.1.152 CKF_CLOCK_ON_TOKEN . . . . .	967
20.171.1.153 CKF_DECRYPT . . . . .	967
20.171.1.154 CKF_DERIVE . . . . .	967
20.171.1.155 CKF_DIGEST . . . . .	968
20.171.1.156 CKF_DONT_BLOCK . . . . .	968
20.171.1.157 CKF_DUAL_CRYPT_Operations . . . . .	968
20.171.1.158 CKF_EC_COMPRESS . . . . .	968
20.171.1.159 CKF_EC_ECPARAMETERS . . . . .	968
20.171.1.160 CKF_EC_F_2M . . . . .	968
20.171.1.161 CKF_EC_F_P . . . . .	968
20.171.1.162 CKF_EC_NAMEDCURVE . . . . .	968
20.171.1.163 CKF_EC_UNCOMPRESS . . . . .	969
20.171.1.164 CKF_ENCRYPT . . . . .	969
20.171.1.165 CKF_ERROR_STATE . . . . .	969
20.171.1.166 CKF_EXCLUDE_CHALLENGE . . . . .	969
20.171.1.167 CKF_EXCLUDE_COUNTER . . . . .	969
20.171.1.168 CKF_EXCLUDE_PIN . . . . .	969

20.171.1.169 CKF_EXCLUDE_TIME . . . . .	969
20.171.1.170 CKF_EXTENSION . . . . .	969
20.171.1.171 CKF_GENERATE . . . . .	970
20.171.1.172 CKF_GENERATE_KEY_PAIR . . . . .	970
20.171.1.173 CKF_HW . . . . .	970
20.171.1.174 CKF_HW_SLOT . . . . .	970
20.171.1.175 CKF_LIBRARY_CANT_CREATE_OS_THREADS . . . . .	970
20.171.1.176 CKF_LOGIN_REQUIRED . . . . .	970
20.171.1.177 CKF_NEXT_OTP . . . . .	970
20.171.1.178 CKF_OS_LOCKING_OK . . . . .	970
20.171.1.179 CKF_PROTECTED_AUTHENTICATION_PATH . . . . .	971
20.171.1.180 CKF_REMOVABLE_DEVICE . . . . .	971
20.171.1.181 CKF_RESTORE_KEY_NOT_NEEDED . . . . .	971
20.171.1.182 CKF_RNG . . . . .	971
20.171.1.183 CKF_RW_SESSION . . . . .	971
20.171.1.184 CKF_SECONDARY_AUTHENTICATION . . . . .	971
20.171.1.185 CKF_SERIAL_SESSION . . . . .	971
20.171.1.186 CKF_SIGN . . . . .	971
20.171.1.187 CKF_SIGN_RECOVER . . . . .	972
20.171.1.188 CKF_SO_PIN_COUNT_LOW . . . . .	972
20.171.1.189 CKF_SO_PIN_FINAL_TRY . . . . .	972
20.171.1.190 CKF_SO_PIN_LOCKED . . . . .	972
20.171.1.191 CKF_SO_PIN_TO_BE_CHANGED . . . . .	972
20.171.1.192 CKF_TOKEN_INITIALIZED . . . . .	972
20.171.1.193 CKF_TOKEN_PRESENT . . . . .	972
20.171.1.194 CKF_UNWRAP . . . . .	972
20.171.1.195 CKF_USER_FRIENDLY_OTP . . . . .	973
20.171.1.196 CKF_USER_PIN_COUNT_LOW . . . . .	973
20.171.1.197 CKF_USER_PIN_FINAL_TRY . . . . .	973
20.171.1.198 CKF_USER_PIN_INITIALIZED . . . . .	973
20.171.1.199 CKF_USER_PIN_LOCKED . . . . .	973
20.171.1.200 CKF_USER_PIN_TO_BE_CHANGED . . . . .	973
20.171.1.201 CKF_VERIFY . . . . .	973
20.171.1.202 CKF_VERIFY_RECOVER . . . . .	973
20.171.1.203 CKF_WRAP . . . . .	974
20.171.1.204 CKF_WRITE_PROTECTED . . . . .	974
20.171.1.205 CKG_MGF1_SHA1 . . . . .	974
20.171.1.206 CKG_MGF1_SHA224 . . . . .	974
20.171.1.207 CKG_MGF1_SHA256 . . . . .	974
20.171.1.208 CKG_MGF1_SHA384 . . . . .	974
20.171.1.209 CKG_MGF1_SHA512 . . . . .	974
20.171.1.210 CKH_CLOCK . . . . .	974

20.171.1.211 CKH_MONOTONIC_COUNTER . . . . .	975
20.171.1.212 CKH_USER_INTERFACE . . . . .	975
20.171.1.213 CKH_VENDOR_DEFINED . . . . .	975
20.171.1.214 CKK_ACTI . . . . .	975
20.171.1.215 CKK_AES . . . . .	975
20.171.1.216 CKK_ARIA . . . . .	975
20.171.1.217 CKK_BATON . . . . .	975
20.171.1.218 CKK_BLOWFISH . . . . .	975
20.171.1.219 CKK_CAMELLIA . . . . .	976
20.171.1.220 CKK_CAST . . . . .	976
20.171.1.221 CKK_CAST128 . . . . .	976
20.171.1.222 CKK_CAST3 . . . . .	976
20.171.1.223 CKK_CAST5 . . . . .	976
20.171.1.224 CKK_CDMF . . . . .	976
20.171.1.225 CKK_DES . . . . .	976
20.171.1.226 CKK_DES2 . . . . .	976
20.171.1.227 CKK_DES3 . . . . .	977
20.171.1.228 CKK_DH . . . . .	977
20.171.1.229 CKK_DSA . . . . .	977
20.171.1.230 CKK_EC . . . . .	977
20.171.1.231 CKK_ECDSA . . . . .	977
20.171.1.232 CKK_GENERIC_SECRET . . . . .	977
20.171.1.233 CKK_GOST28147 . . . . .	977
20.171.1.234 CKK_GOSTR3410 . . . . .	977
20.171.1.235 CKK_GOSTR3411 . . . . .	978
20.171.1.236 CKK_HOTP . . . . .	978
20.171.1.237 CKK_IDEA . . . . .	978
20.171.1.238 CKK_JUNIPER . . . . .	978
20.171.1.239 CKK_KEA . . . . .	978
20.171.1.240 CKK_MD5_HMAC . . . . .	978
20.171.1.241 CKK_RC2 . . . . .	978
20.171.1.242 CKK_RC4 . . . . .	978
20.171.1.243 CKK_RC5 . . . . .	979
20.171.1.244 CKK_RIPEMD128_HMAC . . . . .	979
20.171.1.245 CKK_RIPEMD160_HMAC . . . . .	979
20.171.1.246 CKK_RSA . . . . .	979
20.171.1.247 CKK_SECURID . . . . .	979
20.171.1.248 CKK_SEED . . . . .	979
20.171.1.249 CKK_SHA224_HMAC . . . . .	979
20.171.1.250 CKK_SHA256_HMAC . . . . .	979
20.171.1.251 CKK_SHA384_HMAC . . . . .	980
20.171.1.252 CKK_SHA512_HMAC . . . . .	980

20.171.1.253 CKK_SHA_1_HMAC . . . . .	980
20.171.1.254 CKK_SKIPJACK . . . . .	980
20.171.1.255 CKK_TWOFISH . . . . .	980
20.171.1.256 CKK_VENDOR_DEFINED . . . . .	980
20.171.1.257 CKK_X9_42_DH . . . . .	980
20.171.1.258 CKM_ACTI . . . . .	980
20.171.1.259 CKM_ACTI_KEY_GEN . . . . .	981
20.171.1.260 CKM_AES_CBC . . . . .	981
20.171.1.261 CKM_AES_CBC_ENCRYPT_DATA . . . . .	981
20.171.1.262 CKM_AES_CBC_PAD . . . . .	981
20.171.1.263 CKM_AES_CCM . . . . .	981
20.171.1.264 CKM_AES_CFB1 . . . . .	981
20.171.1.265 CKM_AES_CFB128 . . . . .	981
20.171.1.266 CKM_AES_CFB64 . . . . .	981
20.171.1.267 CKM_AES_CFB8 . . . . .	982
20.171.1.268 CKM_AES_CMAC . . . . .	982
20.171.1.269 CKM_AES_CMAC_GENERAL . . . . .	982
20.171.1.270 CKM_AES_CTR . . . . .	982
20.171.1.271 CKM_AES_CTS . . . . .	982
20.171.1.272 CKM_AES_ECB . . . . .	982
20.171.1.273 CKM_AES_ECB_ENCRYPT_DATA . . . . .	982
20.171.1.274 CKM_AES_GCM . . . . .	982
20.171.1.275 CKM_AES_GMAC . . . . .	983
20.171.1.276 CKM_AES_KEY_GEN . . . . .	983
20.171.1.277 CKM_AES_KEY_WRAP . . . . .	983
20.171.1.278 CKM_AES_KEY_WRAP_PAD . . . . .	983
20.171.1.279 CKM_AES_MAC . . . . .	983
20.171.1.280 CKM_AES_MAC_GENERAL . . . . .	983
20.171.1.281 CKM_AES_OFB . . . . .	983
20.171.1.282 CKM_AES_XCBC_MAC . . . . .	983
20.171.1.283 CKM_AES_XCBC_MAC_96 . . . . .	984
20.171.1.284 CKM_ARIA_CBC . . . . .	984
20.171.1.285 CKM_ARIA_CBC_ENCRYPT_DATA . . . . .	984
20.171.1.286 CKM_ARIA_CBC_PAD . . . . .	984
20.171.1.287 CKM_ARIA_ECB . . . . .	984
20.171.1.288 CKM_ARIA_ECB_ENCRYPT_DATA . . . . .	984
20.171.1.289 CKM_ARIA_KEY_GEN . . . . .	984
20.171.1.290 CKM_ARIA_MAC . . . . .	984
20.171.1.291 CKM_ARIA_MAC_GENERAL . . . . .	985
20.171.1.292 CKM_BATON_CBC128 . . . . .	985
20.171.1.293 CKM_BATON_COUNTER . . . . .	985
20.171.1.294 CKM_BATON_ECB128 . . . . .	985

20.171.1.295 CKM_BATON_ECB96 . . . . .	985
20.171.1.296 CKM_BATON_KEY_GEN . . . . .	985
20.171.1.297 CKM_BATON_SHUFFLE . . . . .	985
20.171.1.298 CKM_BATON_WRAP . . . . .	985
20.171.1.299 CKM_BLOWFISH_CBC . . . . .	986
20.171.1.300 CKM_BLOWFISH_CBC_PAD . . . . .	986
20.171.1.301 CKM_BLOWFISH_KEY_GEN . . . . .	986
20.171.1.302 CKM_CAMELLIA_CBC . . . . .	986
20.171.1.303 CKM_CAMELLIA_CBC_ENCRYPT_DATA . . . . .	986
20.171.1.304 CKM_CAMELLIA_CBC_PAD . . . . .	986
20.171.1.305 CKM_CAMELLIA_CTR . . . . .	986
20.171.1.306 CKM_CAMELLIA_ECB . . . . .	986
20.171.1.307 CKM_CAMELLIA_ECB_ENCRYPT_DATA . . . . .	987
20.171.1.308 CKM_CAMELLIA_KEY_GEN . . . . .	987
20.171.1.309 CKM_CAMELLIA_MAC . . . . .	987
20.171.1.310 CKM_CAMELLIA_MAC_GENERAL . . . . .	987
20.171.1.311 CKM_CAST128_CBC . . . . .	987
20.171.1.312 CKM_CAST128_CBC_PAD . . . . .	987
20.171.1.313 CKM_CAST128_ECB . . . . .	987
20.171.1.314 CKM_CAST128_KEY_GEN . . . . .	987
20.171.1.315 CKM_CAST128_MAC . . . . .	988
20.171.1.316 CKM_CAST128_MAC_GENERAL . . . . .	988
20.171.1.317 CKM_CAST3_CBC . . . . .	988
20.171.1.318 CKM_CAST3_CBC_PAD . . . . .	988
20.171.1.319 CKM_CAST3_ECB . . . . .	988
20.171.1.320 CKM_CAST3_KEY_GEN . . . . .	988
20.171.1.321 CKM_CAST3_MAC . . . . .	988
20.171.1.322 CKM_CAST3_MAC_GENERAL . . . . .	988
20.171.1.323 CKM_CAST5_CBC . . . . .	989
20.171.1.324 CKM_CAST5_CBC_PAD . . . . .	989
20.171.1.325 CKM_CAST5_ECB . . . . .	989
20.171.1.326 CKM_CAST5_KEY_GEN . . . . .	989
20.171.1.327 CKM_CAST5_MAC . . . . .	989
20.171.1.328 CKM_CAST5_MAC_GENERAL . . . . .	989
20.171.1.329 CKM_CAST_CBC . . . . .	989
20.171.1.330 CKM_CAST_CBC_PAD . . . . .	989
20.171.1.331 CKM_CAST_ECB . . . . .	990
20.171.1.332 CKM_CAST_KEY_GEN . . . . .	990
20.171.1.333 CKM_CAST_MAC . . . . .	990
20.171.1.334 CKM_CAST_MAC_GENERAL . . . . .	990
20.171.1.335 CKM_CDMF_CBC . . . . .	990
20.171.1.336 CKM_CDMF_CBC_PAD . . . . .	990



20.171.1.337 CKM_CDMF_ECB . . . . .	990
20.171.1.338 CKM_CDMF_KEY_GEN . . . . .	990
20.171.1.339 CKM_CDMF_MAC . . . . .	991
20.171.1.340 CKM_CDMF_MAC_GENERAL . . . . .	991
20.171.1.341 CKM_CMS_SIG . . . . .	991
20.171.1.342 CKM_CONCATENATE_BASE_AND_DATA . . . . .	991
20.171.1.343 CKM_CONCATENATE_BASE_AND_KEY . . . . .	991
20.171.1.344 CKM_CONCATENATE_DATA_AND_BASE . . . . .	991
20.171.1.345 CKM_DES2_KEY_GEN . . . . .	991
20.171.1.346 CKM_DES3_CBC . . . . .	991
20.171.1.347 CKM_DES3_CBC_ENCRYPT_DATA . . . . .	992
20.171.1.348 CKM_DES3_CBC_PAD . . . . .	992
20.171.1.349 CKM_DES3_CMAC . . . . .	992
20.171.1.350 CKM_DES3_CMAC_GENERAL . . . . .	992
20.171.1.351 CKM_DES3_ECB . . . . .	992
20.171.1.352 CKM_DES3_ECB_ENCRYPT_DATA . . . . .	992
20.171.1.353 CKM_DES3_KEY_GEN . . . . .	992
20.171.1.354 CKM_DES3_MAC . . . . .	992
20.171.1.355 CKM_DES3_MAC_GENERAL . . . . .	993
20.171.1.356 CKM_DES_CBC . . . . .	993
20.171.1.357 CKM_DES_CBC_ENCRYPT_DATA . . . . .	993
20.171.1.358 CKM_DES_CBC_PAD . . . . .	993
20.171.1.359 CKM_DES_CFB64 . . . . .	993
20.171.1.360 CKM_DES_CFB8 . . . . .	993
20.171.1.361 CKM_DES_ECB . . . . .	993
20.171.1.362 CKM_DES_ECB_ENCRYPT_DATA . . . . .	993
20.171.1.363 CKM_DES_KEY_GEN . . . . .	994
20.171.1.364 CKM_DES_MAC . . . . .	994
20.171.1.365 CKM_DES_MAC_GENERAL . . . . .	994
20.171.1.366 CKM_DES_OFB64 . . . . .	994
20.171.1.367 CKM_DES_OFB8 . . . . .	994
20.171.1.368 CKM_DH_PKCS_DERIVE . . . . .	994
20.171.1.369 CKM_DH_PKCS_KEY_PAIR_GEN . . . . .	994
20.171.1.370 CKM_DH_PKCS_PARAMETER_GEN . . . . .	994
20.171.1.371 CKM_DSA . . . . .	995
20.171.1.372 CKM_DSA_KEY_PAIR_GEN . . . . .	995
20.171.1.373 CKM_DSA_PARAMETER_GEN . . . . .	995
20.171.1.374 CKM_DSA_PROBABLISTIC_PARAMETER_GEN . . . . .	995
20.171.1.375 CKM_DSA_SHA1 . . . . .	995
20.171.1.376 CKM_DSA_SHA224 . . . . .	995
20.171.1.377 CKM_DSA_SHA256 . . . . .	995
20.171.1.378 CKM_DSA_SHA384 . . . . .	995

20.171.1.379 CKM_DSA_SHA512 . . . . .	996
20.171.1.380 CKM_DSA_SHAKE_TAYLOR_PARAMETER_GEN . . . . .	996
20.171.1.381 CKM_EC_KEY_PAIR_GEN . . . . .	996
20.171.1.382 CKM_ECDH1_COFACTOR_DERIVE . . . . .	996
20.171.1.383 CKM_ECDH1_DERIVE . . . . .	996
20.171.1.384 CKM_ECDH_AES_KEY_WRAP . . . . .	996
20.171.1.385 CKM_ECDSA . . . . .	996
20.171.1.386 CKM_ECDSA_KEY_PAIR_GEN . . . . .	996
20.171.1.387 CKM_ECDSA_SHA1 . . . . .	997
20.171.1.388 CKM_ECDSA_SHA224 . . . . .	997
20.171.1.389 CKM_ECDSA_SHA256 . . . . .	997
20.171.1.390 CKM_ECDSA_SHA384 . . . . .	997
20.171.1.391 CKM_ECDSA_SHA512 . . . . .	997
20.171.1.392 CKM_ECMQV_DERIVE . . . . .	997
20.171.1.393 CKM_EXTRACT_KEY_FROM_KEY . . . . .	997
20.171.1.394 CKM_FASTHASH . . . . .	997
20.171.1.395 CKM_FORTEZZA_TIMESTAMP . . . . .	998
20.171.1.396 CKM_GENERIC_SECRET_KEY_GEN . . . . .	998
20.171.1.397 CKM_GOST28147 . . . . .	998
20.171.1.398 CKM_GOST28147_ECB . . . . .	998
20.171.1.399 CKM_GOST28147_KEY_GEN . . . . .	998
20.171.1.400 CKM_GOST28147_KEY_WRAP . . . . .	998
20.171.1.401 CKM_GOST28147_MAC . . . . .	998
20.171.1.402 CKM_GOSTR3410 . . . . .	998
20.171.1.403 CKM_GOSTR3410_DERIVE . . . . .	999
20.171.1.404 CKM_GOSTR3410_KEY_PAIR_GEN . . . . .	999
20.171.1.405 CKM_GOSTR3410_KEY_WRAP . . . . .	999
20.171.1.406 CKM_GOSTR3410_WITH_GOSTR3411 . . . . .	999
20.171.1.407 CKM_GOSTR3411 . . . . .	999
20.171.1.408 CKM_GOSTR3411_HMAC . . . . .	999
20.171.1.409 CKM_HOTP . . . . .	999
20.171.1.410 CKM_HOTP_KEY_GEN . . . . .	999
20.171.1.411 CKM_IDEA_CBC . . . . .	1000
20.171.1.412 CKM_IDEA_CBC_PAD . . . . .	1000
20.171.1.413 CKM_IDEA_ECB . . . . .	1000
20.171.1.414 CKM_IDEA_KEY_GEN . . . . .	1000
20.171.1.415 CKM_IDEA_MAC . . . . .	1000
20.171.1.416 CKM_IDEA_MAC_GENERAL . . . . .	1000
20.171.1.417 CKM_JUNIPER_CBC128 . . . . .	1000
20.171.1.418 CKM_JUNIPER_COUNTER . . . . .	1000
20.171.1.419 CKM_JUNIPER_ECB128 . . . . .	1001
20.171.1.420 CKM_JUNIPER_KEY_GEN . . . . .	1001

20.171.1.421 CKM_JUNIPER_SHUFFLE . . . . .	1001
20.171.1.422 CKM_JUNIPER_WRAP . . . . .	1001
20.171.1.423 CKM_KEA_DERIVE . . . . .	1001
20.171.1.424 CKM_KEA_KEY_DERIVE . . . . .	1001
20.171.1.425 CKM_KEA_KEY_PAIR_GEN . . . . .	1001
20.171.1.426 CKM_KEY_WRAP_LYNKS . . . . .	1001
20.171.1.427 CKM_KEY_WRAP_SET_OAEP . . . . .	1002
20.171.1.428 CKM_KIP_DERIVE . . . . .	1002
20.171.1.429 CKM_KIP_MAC . . . . .	1002
20.171.1.430 CKM_KIP_WRAP . . . . .	1002
20.171.1.431 CKM_MD2 . . . . .	1002
20.171.1.432 CKM_MD2_HMAC . . . . .	1002
20.171.1.433 CKM_MD2_HMAC_GENERAL . . . . .	1002
20.171.1.434 CKM_MD2_KEY_DERIVATION . . . . .	1002
20.171.1.435 CKM_MD2_RSA_PKCS . . . . .	1003
20.171.1.436 CKM_MD5 . . . . .	1003
20.171.1.437 CKM_MD5_HMAC . . . . .	1003
20.171.1.438 CKM_MD5_HMAC_GENERAL . . . . .	1003
20.171.1.439 CKM_MD5_KEY_DERIVATION . . . . .	1003
20.171.1.440 CKM_MD5_RSA_PKCS . . . . .	1003
20.171.1.441 CKM_PBA_SHA1_WITH_SHA1_HMAC . . . . .	1003
20.171.1.442 CKM_PBE_MD2_DES_CBC . . . . .	1003
20.171.1.443 CKM_PBE_MD5_CAST128_CBC . . . . .	1004
20.171.1.444 CKM_PBE_MD5_CAST3_CBC . . . . .	1004
20.171.1.445 CKM_PBE_MD5_CAST5_CBC . . . . .	1004
20.171.1.446 CKM_PBE_MD5_CAST_CBC . . . . .	1004
20.171.1.447 CKM_PBE_MD5_DES_CBC . . . . .	1004
20.171.1.448 CKM_PBE_SHA1_CAST128_CBC . . . . .	1004
20.171.1.449 CKM_PBE_SHA1_CAST5_CBC . . . . .	1004
20.171.1.450 CKM_PBE_SHA1_DES2_EDE_CBC . . . . .	1004
20.171.1.451 CKM_PBE_SHA1_DES3_EDE_CBC . . . . .	1005
20.171.1.452 CKM_PBE_SHA1_RC2_128_CBC . . . . .	1005
20.171.1.453 CKM_PBE_SHA1_RC2_40_CBC . . . . .	1005
20.171.1.454 CKM_PBE_SHA1_RC4_128 . . . . .	1005
20.171.1.455 CKM_PBE_SHA1_RC4_40 . . . . .	1005
20.171.1.456 CKM_PKCS5_PBKD2 . . . . .	1005
20.171.1.457 CKM_RC2_CBC . . . . .	1005
20.171.1.458 CKM_RC2_CBC_PAD . . . . .	1005
20.171.1.459 CKM_RC2_ECB . . . . .	1006
20.171.1.460 CKM_RC2_KEY_GEN . . . . .	1006
20.171.1.461 CKM_RC2_MAC . . . . .	1006
20.171.1.462 CKM_RC2_MAC_GENERAL . . . . .	1006

20.171.1.463 CKM_RC4 . . . . .	1006
20.171.1.464 CKM_RC4_KEY_GEN . . . . .	1006
20.171.1.465 CKM_RC5_CBC . . . . .	1006
20.171.1.466 CKM_RC5_CBC_PAD . . . . .	1006
20.171.1.467 CKM_RC5_ECB . . . . .	1007
20.171.1.468 CKM_RC5_KEY_GEN . . . . .	1007
20.171.1.469 CKM_RC5_MAC . . . . .	1007
20.171.1.470 CKM_RC5_MAC_GENERAL . . . . .	1007
20.171.1.471 CKM_RIPEMD128 . . . . .	1007
20.171.1.472 CKM_RIPEMD128_HMAC . . . . .	1007
20.171.1.473 CKM_RIPEMD128_HMAC_GENERAL . . . . .	1007
20.171.1.474 CKM_RIPEMD128_RSA_PKCS . . . . .	1007
20.171.1.475 CKM_RIPEMD160 . . . . .	1008
20.171.1.476 CKM_RIPEMD160_HMAC . . . . .	1008
20.171.1.477 CKM_RIPEMD160_HMAC_GENERAL . . . . .	1008
20.171.1.478 CKM_RIPEMD160_RSA_PKCS . . . . .	1008
20.171.1.479 CKM_RSA_9796 . . . . .	1008
20.171.1.480 CKM_RSA_AES_KEY_WRAP . . . . .	1008
20.171.1.481 CKM_RSA_PKCS . . . . .	1008
20.171.1.482 CKM_RSA_PKCS_KEY_PAIR_GEN . . . . .	1008
20.171.1.483 CKM_RSA_PKCS_OAEP . . . . .	1009
20.171.1.484 CKM_RSA_PKCS_OAEP_TPM_1_1 . . . . .	1009
20.171.1.485 CKM_RSA_PKCS_PSS . . . . .	1009
20.171.1.486 CKM_RSA_PKCS_TPM_1_1 . . . . .	1009
20.171.1.487 CKM_RSA_X9_31 . . . . .	1009
20.171.1.488 CKM_RSA_X9_31_KEY_PAIR_GEN . . . . .	1009
20.171.1.489 CKM_RSA_X_509 . . . . .	1009
20.171.1.490 CKM_SECURID . . . . .	1009
20.171.1.491 CKM_SECURID_KEY_GEN . . . . .	1010
20.171.1.492 CKM_SEED_CBC . . . . .	1010
20.171.1.493 CKM_SEED_CBC_ENCRYPT_DATA . . . . .	1010
20.171.1.494 CKM_SEED_CBC_PAD . . . . .	1010
20.171.1.495 CKM_SEED_ECB . . . . .	1010
20.171.1.496 CKM_SEED_ECB_ENCRYPT_DATA . . . . .	1010
20.171.1.497 CKM_SEED_KEY_GEN . . . . .	1010
20.171.1.498 CKM_SEED_MAC . . . . .	1010
20.171.1.499 CKM_SEED_MAC_GENERAL . . . . .	1011
20.171.1.500 CKM_SHA1_KEY_DERIVATION . . . . .	1011
20.171.1.501 CKM_SHA1_RSA_PKCS . . . . .	1011
20.171.1.502 CKM_SHA1_RSA_PKCS_PSS . . . . .	1011
20.171.1.503 CKM_SHA1_RSA_X9_31 . . . . .	1011
20.171.1.504 CKM_SHA224 . . . . .	1011

20.171.1.505 CKM_SHA224_HMAC . . . . .	1011
20.171.1.506 CKM_SHA224_HMAC_GENERAL . . . . .	1011
20.171.1.507 CKM_SHA224_KEY_DERIVATION . . . . .	1012
20.171.1.508 CKM_SHA224_RSA_PKCS . . . . .	1012
20.171.1.509 CKM_SHA224_RSA_PKCS_PSS . . . . .	1012
20.171.1.510 CKM_SHA256 . . . . .	1012
20.171.1.511 CKM_SHA256_HMAC . . . . .	1012
20.171.1.512 CKM_SHA256_HMAC_GENERAL . . . . .	1012
20.171.1.513 CKM_SHA256_KEY_DERIVATION . . . . .	1012
20.171.1.514 CKM_SHA256_RSA_PKCS . . . . .	1012
20.171.1.515 CKM_SHA256_RSA_PKCS_PSS . . . . .	1013
20.171.1.516 CKM_SHA384 . . . . .	1013
20.171.1.517 CKM_SHA384_HMAC . . . . .	1013
20.171.1.518 CKM_SHA384_HMAC_GENERAL . . . . .	1013
20.171.1.519 CKM_SHA384_KEY_DERIVATION . . . . .	1013
20.171.1.520 CKM_SHA384_RSA_PKCS . . . . .	1013
20.171.1.521 CKM_SHA384_RSA_PKCS_PSS . . . . .	1013
20.171.1.522 CKM_SHA512 . . . . .	1013
20.171.1.523 CKM_SHA512_224 . . . . .	1014
20.171.1.524 CKM_SHA512_224_HMAC . . . . .	1014
20.171.1.525 CKM_SHA512_224_HMAC_GENERAL . . . . .	1014
20.171.1.526 CKM_SHA512_224_KEY_DERIVATION . . . . .	1014
20.171.1.527 CKM_SHA512_256 . . . . .	1014
20.171.1.528 CKM_SHA512_256_HMAC . . . . .	1014
20.171.1.529 CKM_SHA512_256_HMAC_GENERAL . . . . .	1014
20.171.1.530 CKM_SHA512_256_KEY_DERIVATION . . . . .	1014
20.171.1.531 CKM_SHA512_HMAC . . . . .	1015
20.171.1.532 CKM_SHA512_HMAC_GENERAL . . . . .	1015
20.171.1.533 CKM_SHA512_KEY_DERIVATION . . . . .	1015
20.171.1.534 CKM_SHA512_RSA_PKCS . . . . .	1015
20.171.1.535 CKM_SHA512_RSA_PKCS_PSS . . . . .	1015
20.171.1.536 CKM_SHA512_T . . . . .	1015
20.171.1.537 CKM_SHA512_T_HMAC . . . . .	1015
20.171.1.538 CKM_SHA512_T_HMAC_GENERAL . . . . .	1015
20.171.1.539 CKM_SHA512_T_KEY_DERIVATION . . . . .	1016
20.171.1.540 CKM_SHA_1 . . . . .	1016
20.171.1.541 CKM_SHA_1_HMAC . . . . .	1016
20.171.1.542 CKM_SHA_1_HMAC_GENERAL . . . . .	1016
20.171.1.543 CKM_SKIPJACK_CBC64 . . . . .	1016
20.171.1.544 CKM_SKIPJACK_CFB16 . . . . .	1016
20.171.1.545 CKM_SKIPJACK_CFB32 . . . . .	1016
20.171.1.546 CKM_SKIPJACK_CFB64 . . . . .	1016

20.171.1.547 CKM_SKIPJACK_CFB8 . . . . .	1017
20.171.1.548 CKM_SKIPJACK_ECB64 . . . . .	1017
20.171.1.549 CKM_SKIPJACK_KEY_GEN . . . . .	1017
20.171.1.550 CKM_SKIPJACK_OFB64 . . . . .	1017
20.171.1.551 CKM_SKIPJACK_PRIVATE_WRAP . . . . .	1017
20.171.1.552 CKM_SKIPJACK_RELAYX . . . . .	1017
20.171.1.553 CKM_SKIPJACK_WRAP . . . . .	1017
20.171.1.554 CKM_SSL3_KEY_AND_MAC_DERIVE . . . . .	1017
20.171.1.555 CKM_SSL3_MASTER_KEY_DERIVE . . . . .	1018
20.171.1.556 CKM_SSL3_MASTER_KEY_DERIVE_DH . . . . .	1018
20.171.1.557 CKM_SSL3_MD5_MAC . . . . .	1018
20.171.1.558 CKM_SSL3_PRE_MASTER_KEY_GEN . . . . .	1018
20.171.1.559 CKM_SSL3_SHA1_MAC . . . . .	1018
20.171.1.560 CKM_TLS10_MAC_CLIENT . . . . .	1018
20.171.1.561 CKM_TLS10_MAC_SERVER . . . . .	1018
20.171.1.562 CKM_TLS12_KDF . . . . .	1018
20.171.1.563 CKM_TLS12_KEY_AND_MAC_DERIVE . . . . .	1019
20.171.1.564 CKM_TLS12_KEY_SAFE_DERIVE . . . . .	1019
20.171.1.565 CKM_TLS12_MAC . . . . .	1019
20.171.1.566 CKM_TLS12_MASTER_KEY_DERIVE . . . . .	1019
20.171.1.567 CKM_TLS12_MASTER_KEY_DERIVE_DH . . . . .	1019
20.171.1.568 CKM_TLS_KDF . . . . .	1019
20.171.1.569 CKM_TLS_KEY_AND_MAC_DERIVE . . . . .	1019
20.171.1.570 CKM_TLS_MAC . . . . .	1019
20.171.1.571 CKM_TLS_MASTER_KEY_DERIVE . . . . .	1020
20.171.1.572 CKM_TLS_MASTER_KEY_DERIVE_DH . . . . .	1020
20.171.1.573 CKM_TLS_PRE_MASTER_KEY_GEN . . . . .	1020
20.171.1.574 CKM_TLS_PRF . . . . .	1020
20.171.1.575 CKM_TWOFISH_CBC . . . . .	1020
20.171.1.576 CKM_TWOFISH_CBC_PAD . . . . .	1020
20.171.1.577 CKM_TWOFISH_KEY_GEN . . . . .	1020
20.171.1.578 CKM_VENDOR_DEFINED . . . . .	1020
20.171.1.579 CKM_WTLS_CLIENT_KEY_AND_MAC_DERIVE . . . . .	1021
20.171.1.580 CKM_WTLS_MASTER_KEY_DERIVE . . . . .	1021
20.171.1.581 CKM_WTLS_MASTER_KEY_DERIVE_DH_ECC . . . . .	1021
20.171.1.582 CKM_WTLS_PRE_MASTER_KEY_GEN . . . . .	1021
20.171.1.583 CKM_WTLS_PRF . . . . .	1021
20.171.1.584 CKM_WTLS_SERVER_KEY_AND_MAC_DERIVE . . . . .	1021
20.171.1.585 CKM_X9_42_DH_DERIVE . . . . .	1021
20.171.1.586 CKM_X9_42_DH_HYBRID_DERIVE . . . . .	1021
20.171.1.587 CKM_X9_42_DH_KEY_PAIR_GEN . . . . .	1022
20.171.1.588 CKM_X9_42_DH_PARAMETER_GEN . . . . .	1022

20.171.1.589 CKM_X9_42_MQV_DERIVE . . . . .	1022
20.171.1.590 CKM_XOR_BASE_AND_DATA . . . . .	1022
20.171.1.591 CKN_OTP_CHANGED . . . . .	1022
20.171.1.592 CKN_SURRENDER . . . . .	1022
20.171.1.593 CKO_CERTIFICATE . . . . .	1022
20.171.1.594 CKO_DATA . . . . .	1022
20.171.1.595 CKO_DOMAIN_PARAMETERS . . . . .	1023
20.171.1.596 CKO_HW_FEATURE . . . . .	1023
20.171.1.597 CKO_MECHANISM . . . . .	1023
20.171.1.598 CKO_OTP_KEY . . . . .	1023
20.171.1.599 CKO_PRIVATE_KEY . . . . .	1023
20.171.1.600 CKO_PUBLIC_KEY . . . . .	1023
20.171.1.601 CKO_SECRET_KEY . . . . .	1023
20.171.1.602 CKO_VENDOR_DEFINED . . . . .	1023
20.171.1.603 CKP_PKCS5_PBKD2_HMAC_GOSTR3411 . . . . .	1024
20.171.1.604 CKP_PKCS5_PBKD2_HMAC_SHA1 . . . . .	1024
20.171.1.605 CKP_PKCS5_PBKD2_HMAC_SHA224 . . . . .	1024
20.171.1.606 CKP_PKCS5_PBKD2_HMAC_SHA256 . . . . .	1024
20.171.1.607 CKP_PKCS5_PBKD2_HMAC_SHA384 . . . . .	1024
20.171.1.608 CKP_PKCS5_PBKD2_HMAC_SHA512 . . . . .	1024
20.171.1.609 CKP_PKCS5_PBKD2_HMAC_SHA512_224 . . . . .	1024
20.171.1.610 CKP_PKCS5_PBKD2_HMAC_SHA512_256 . . . . .	1024
20.171.1.611 CKR_ACTION_PROHIBITED . . . . .	1025
20.171.1.612 CKR_ARGUMENTS_BAD . . . . .	1025
20.171.1.613 CKR_ATTRIBUTE_READ_ONLY . . . . .	1025
20.171.1.614 CKR_ATTRIBUTE_SENSITIVE . . . . .	1025
20.171.1.615 CKR_ATTRIBUTE_TYPE_INVALID . . . . .	1025
20.171.1.616 CKR_ATTRIBUTE_VALUE_INVALID . . . . .	1025
20.171.1.617 CKR_BUFFER_TOO_SMALL . . . . .	1025
20.171.1.618 CKR_CANCEL . . . . .	1025
20.171.1.619 CKR_CANT_LOCK . . . . .	1026
20.171.1.620 CKR_CRYPTOKI_ALREADY_INITIALIZED . . . . .	1026
20.171.1.621 CKR_CRYPTOKI_NOT_INITIALIZED . . . . .	1026
20.171.1.622 CKR_CURVE_NOT_SUPPORTED . . . . .	1026
20.171.1.623 CKR_DATA_INVALID . . . . .	1026
20.171.1.624 CKR_DATA_LEN_RANGE . . . . .	1026
20.171.1.625 CKR_DEVICE_ERROR . . . . .	1026
20.171.1.626 CKR_DEVICE_MEMORY . . . . .	1026
20.171.1.627 CKR_DEVICE_REMOVED . . . . .	1027
20.171.1.628 CKR_DOMAIN_PARAMS_INVALID . . . . .	1027
20.171.1.629 CKR_ENCRYPTED_DATA_INVALID . . . . .	1027
20.171.1.630 CKR_ENCRYPTED_DATA_LEN_RANGE . . . . .	1027

---

20.171.1.631 CKR_EXCEEDED_MAX_ITERATIONS . . . . .	1027
20.171.1.632 CKR_FIPS_SELF_TEST_FAILED . . . . .	1027
20.171.1.633 CKR_FUNCTION_CANCELED . . . . .	1027
20.171.1.634 CKR_FUNCTION_FAILED . . . . .	1027
20.171.1.635 CKR_FUNCTION_NOT_PARALLEL . . . . .	1028
20.171.1.636 CKR_FUNCTION_NOT_SUPPORTED . . . . .	1028
20.171.1.637 CKR_FUNCTION_REJECTED . . . . .	1028
20.171.1.638 CKR_GENERAL_ERROR . . . . .	1028
20.171.1.639 CKR_HOST_MEMORY . . . . .	1028
20.171.1.640 CKR_INFORMATION_SENSITIVE . . . . .	1028
20.171.1.641 CKR_KEY_CHANGED . . . . .	1028
20.171.1.642 CKR_KEY_FUNCTION_NOT_PERMITTED . . . . .	1028
20.171.1.643 CKR_KEY_HANDLE_INVALID . . . . .	1029
20.171.1.644 CKR_KEY_INDIGESTIBLE . . . . .	1029
20.171.1.645 CKR_KEY_NEEDED . . . . .	1029
20.171.1.646 CKR_KEY_NOT_NEEDED . . . . .	1029
20.171.1.647 CKR_KEY_NOT_WRAPABLE . . . . .	1029
20.171.1.648 CKR_KEY_SIZE_RANGE . . . . .	1029
20.171.1.649 CKR_KEY_TYPE_INCONSISTENT . . . . .	1029
20.171.1.650 CKR_KEY_UNEXTRACTABLE . . . . .	1029
20.171.1.651 CKR_LIBRARY_LOAD_FAILED . . . . .	1030
20.171.1.652 CKR_MECHANISM_INVALID . . . . .	1030
20.171.1.653 CKR_MECHANISM_PARAM_INVALID . . . . .	1030
20.171.1.654 CKR_MUTEX_BAD . . . . .	1030
20.171.1.655 CKR_MUTEX_NOT_LOCKED . . . . .	1030
20.171.1.656 CKR_NEED_TO_CREATE_THREADS . . . . .	1030
20.171.1.657 CKR_NEW_PIN_MODE . . . . .	1030
20.171.1.658 CKR_NEXT_OTP . . . . .	1030
20.171.1.659 CKR_NO_EVENT . . . . .	1031
20.171.1.660 CKR_OBJECT_HANDLE_INVALID . . . . .	1031
20.171.1.661 CKR_OK . . . . .	1031
20.171.1.662 CKR_OPERATION_ACTIVE . . . . .	1031
20.171.1.663 CKR_OPERATION_NOT_INITIALIZED . . . . .	1031
20.171.1.664 CKR_PIN_EXPIRED . . . . .	1031
20.171.1.665 CKR_PIN_INCORRECT . . . . .	1031
20.171.1.666 CKR_PIN_INVALID . . . . .	1031
20.171.1.667 CKR_PIN_LEN_RANGE . . . . .	1032
20.171.1.668 CKR_PIN_LOCKED . . . . .	1032
20.171.1.669 CKR_PIN_TOO_WEAK . . . . .	1032
20.171.1.670 CKR_PUBLIC_KEY_INVALID . . . . .	1032
20.171.1.671 CKR_RANDOM_NO_RNG . . . . .	1032
20.171.1.672 CKR_RANDOM_SEED_NOT_SUPPORTED . . . . .	1032



20.171.1.673 CKR_SAVED_STATE_INVALID . . . . .	1032
20.171.1.674 CKR_SESSION_CLOSED . . . . .	1032
20.171.1.675 CKR_SESSION_COUNT . . . . .	1033
20.171.1.676 CKR_SESSION_EXISTS . . . . .	1033
20.171.1.677 CKR_SESSION_HANDLE_INVALID . . . . .	1033
20.171.1.678 CKR_SESSION_PARALLEL_NOT_SUPPORTED . . . . .	1033
20.171.1.679 CKR_SESSION_READ_ONLY . . . . .	1033
20.171.1.680 CKR_SESSION_READ_ONLY_EXISTS . . . . .	1033
20.171.1.681 CKR_SESSION_READ_WRITE_SO_EXISTS . . . . .	1033
20.171.1.682 CKR_SIGNATURE_INVALID . . . . .	1033
20.171.1.683 CKR_SIGNATURE_LEN_RANGE . . . . .	1034
20.171.1.684 CKR_SLOT_ID_INVALID . . . . .	1034
20.171.1.685 CKR_STATE_UNSAVEABLE . . . . .	1034
20.171.1.686 CKR_TEMPLATE_INCOMPLETE . . . . .	1034
20.171.1.687 CKR_TEMPLATE_INCONSISTENT . . . . .	1034
20.171.1.688 CKR_TOKEN_NOT_PRESENT . . . . .	1034
20.171.1.689 CKR_TOKEN_NOT_RECOGNIZED . . . . .	1034
20.171.1.690 CKR_TOKEN_WRITE_PROTECTED . . . . .	1034
20.171.1.691 CKR_UNWRAPPING_KEY_HANDLE_INVALID . . . . .	1035
20.171.1.692 CKR_UNWRAPPING_KEY_SIZE_RANGE . . . . .	1035
20.171.1.693 CKR_UNWRAPPING_KEY_TYPE_INCONSISTENT . . . . .	1035
20.171.1.694 CKR_USER_ALREADY_LOGGED_IN . . . . .	1035
20.171.1.695 CKR_USER_ANOTHER_ALREADY_LOGGED_IN . . . . .	1035
20.171.1.696 CKR_USER_NOT_LOGGED_IN . . . . .	1035
20.171.1.697 CKR_USER_PIN_NOT_INITIALIZED . . . . .	1035
20.171.1.698 CKR_USER_TOO_MANY_TYPES . . . . .	1035
20.171.1.699 CKR_USER_TYPE_INVALID . . . . .	1036
20.171.1.700 CKR_VENDOR_DEFINED . . . . .	1036
20.171.1.701 CKR_WRAPPED_KEY_INVALID . . . . .	1036
20.171.1.702 CKR_WRAPPED_KEY_LEN_RANGE . . . . .	1036
20.171.1.703 CKR_WRAPPING_KEY_HANDLE_INVALID . . . . .	1036
20.171.1.704 CKR_WRAPPING_KEY_SIZE_RANGE . . . . .	1036
20.171.1.705 CKR_WRAPPING_KEY_TYPE_INCONSISTENT . . . . .	1036
20.171.1.706 CKS_RO_PUBLIC_SESSION . . . . .	1036
20.171.1.707 CKS_RO_USER_FUNCTIONS . . . . .	1037
20.171.1.708 CKS_RW_PUBLIC_SESSION . . . . .	1037
20.171.1.709 CKS_RW_SO_FUNCTIONS . . . . .	1037
20.171.1.710 CKS_RW_USER_FUNCTIONS . . . . .	1037
20.171.1.711 CKU_CONTEXT_SPECIFIC . . . . .	1037
20.171.1.712 CKU_SO . . . . .	1037
20.171.1.713 CKU_USER . . . . .	1037
20.171.1.714 CKZ_DATA_SPECIFIED . . . . .	1037

20.171.1.715 CKZ_SALT_SPECIFIED . . . . .	1038
20.171.1.716 CRYPTOKI_VERSION_AMENDMENT . . . . .	1038
20.171.1.717 CRYPTOKI_VERSION_MAJOR . . . . .	1038
20.171.1.718 CRYPTOKI_VERSION_MINOR . . . . .	1038
20.171.1.719 FALSE . . . . .	1038
20.171.1.720 TRUE . . . . .	1038
20.171.2 Typedef Documentation . . . . .	1038
20.171.2.1 CK_AES_CBC_ENCRYPT_DATA_PARAMS . . . . .	1038
20.171.2.2 CK_AES_CBC_ENCRYPT_DATA_PARAMS_PTR . . . . .	1039
20.171.2.3 CK_AES_CCM_PARAMS . . . . .	1039
20.171.2.4 CK_AES_CCM_PARAMS_PTR . . . . .	1039
20.171.2.5 CK_AES_CTR_PARAMS . . . . .	1039
20.171.2.6 CK_AES_CTR_PARAMS_PTR . . . . .	1039
20.171.2.7 CK_AES_GCM_PARAMS . . . . .	1039
20.171.2.8 CK_AES_GCM_PARAMS_PTR . . . . .	1039
20.171.2.9 CK_ARIA_CBC_ENCRYPT_DATA_PARAMS . . . . .	1039
20.171.2.10 CK_ARIA_CBC_ENCRYPT_DATA_PARAMS_PTR . . . . .	1040
20.171.2.11 CK_ATTRIBUTE . . . . .	1040
20.171.2.12 CK_ATTRIBUTE_PTR . . . . .	1040
20.171.2.13 CK_ATTRIBUTE_TYPE . . . . .	1040
20.171.2.14 CK_BBOOL . . . . .	1040
20.171.2.15 CK_BYTE . . . . .	1040
20.171.2.16 CK_BYTE_PTR . . . . .	1040
20.171.2.17 CK_C_INITIALIZE_ARGS . . . . .	1040
20.171.2.18 CK_C_INITIALIZE_ARGS_PTR . . . . .	1041
20.171.2.19 CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS . . . . .	1041
20.171.2.20 CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS_PTR . . . . .	1041
20.171.2.21 CK_CAMELLIA_CTR_PARAMS . . . . .	1041
20.171.2.22 CK_CAMELLIA_CTR_PARAMS_PTR . . . . .	1041
20.171.2.23 CK_CCM_PARAMS . . . . .	1041
20.171.2.24 CK_CCM_PARAMS_PTR . . . . .	1041
20.171.2.25 CK_CERTIFICATE_CATEGORY . . . . .	1041
20.171.2.26 CK_CERTIFICATE_TYPE . . . . .	1042
20.171.2.27 CK_CHAR . . . . .	1042
20.171.2.28 CK_CHAR_PTR . . . . .	1042
20.171.2.29 CK_CMS_SIG_PARAMS . . . . .	1042
20.171.2.30 CK_CMS_SIG_PARAMS_PTR . . . . .	1042
20.171.2.31 CK_DATE . . . . .	1042
20.171.2.32 CK_DES_CBC_ENCRYPT_DATA_PARAMS . . . . .	1042
20.171.2.33 CK_DES_CBC_ENCRYPT_DATA_PARAMS_PTR . . . . .	1042
20.171.2.34 CK_DSA_PARAMETER_GEN_PARAM . . . . .	1043
20.171.2.35 CK_DSA_PARAMETER_GEN_PARAM_PTR . . . . .	1043

20.171.2.36 CK_EC_KDF_TYPE . . . . .	1043
20.171.2.37 CK_ECDH1_DERIVE_PARAMS . . . . .	1043
20.171.2.38 CK_ECDH1_DERIVE_PARAMS_PTR . . . . .	1043
20.171.2.39 CK_ECDH2_DERIVE_PARAMS . . . . .	1043
20.171.2.40 CK_ECDH2_DERIVE_PARAMS_PTR . . . . .	1043
20.171.2.41 CK_ECDH_AES_KEY_WRAP_PARAMS . . . . .	1043
20.171.2.42 CK_ECDH_AES_KEY_WRAP_PARAMS_PTR . . . . .	1044
20.171.2.43 CK_ECMQV_DERIVE_PARAMS . . . . .	1044
20.171.2.44 CK_ECMQV_DERIVE_PARAMS_PTR . . . . .	1044
20.171.2.45 CK_EXTRACT_PARAMS . . . . .	1044
20.171.2.46 CK_EXTRACT_PARAMS_PTR . . . . .	1044
20.171.2.47 CK_FLAGS . . . . .	1044
20.171.2.48 CK_FUNCTION_LIST . . . . .	1044
20.171.2.49 CK_FUNCTION_LIST_PTR . . . . .	1044
20.171.2.50 CK_FUNCTION_LIST_PTR_PTR . . . . .	1045
20.171.2.51 CK_GCM_PARAMS . . . . .	1045
20.171.2.52 CK_GCM_PARAMS_PTR . . . . .	1045
20.171.2.53 CK_GOSTR3410_DERIVE_PARAMS . . . . .	1045
20.171.2.54 CK_GOSTR3410_DERIVE_PARAMS_PTR . . . . .	1045
20.171.2.55 CK_GOSTR3410_KEY_WRAP_PARAMS . . . . .	1045
20.171.2.56 CK_GOSTR3410_KEY_WRAP_PARAMS_PTR . . . . .	1045
20.171.2.57 CK_HW_FEATURE_TYPE . . . . .	1045
20.171.2.58 CK_INFO . . . . .	1046
20.171.2.59 CK_INFO_PTR . . . . .	1046
20.171.2.60 CK_JAVA_MIDP_SECURITY_DOMAIN . . . . .	1046
20.171.2.61 CK_KEA_DERIVE_PARAMS . . . . .	1046
20.171.2.62 CK_KEA_DERIVE_PARAMS_PTR . . . . .	1046
20.171.2.63 CK_KEY_DERIVATION_STRING_DATA . . . . .	1046
20.171.2.64 CK_KEY_DERIVATION_STRING_DATA_PTR . . . . .	1046
20.171.2.65 CK_KEY_TYPE . . . . .	1046
20.171.2.66 CK_KEY_WRAP_SET_OAEP_PARAMS . . . . .	1047
20.171.2.67 CK_KEY_WRAP_SET_OAEP_PARAMS_PTR . . . . .	1047
20.171.2.68 CK_KIP_PARAMS . . . . .	1047
20.171.2.69 CK_KIP_PARAMS_PTR . . . . .	1047
20.171.2.70 CK_LONG . . . . .	1047
20.171.2.71 CK_MAC_GENERAL_PARAMS . . . . .	1047
20.171.2.72 CK_MAC_GENERAL_PARAMS_PTR . . . . .	1047
20.171.2.73 CK_MECHANISM . . . . .	1047
20.171.2.74 CK_MECHANISM_INFO . . . . .	1048
20.171.2.75 CK_MECHANISM_INFO_PTR . . . . .	1048
20.171.2.76 CK_MECHANISM_PTR . . . . .	1048
20.171.2.77 CK_MECHANISM_TYPE . . . . .	1048

20.171.2.78 CK_MECHANISM_TYPE_PTR . . . . .	1048
20.171.2.79 CK_NOTIFICATION . . . . .	1048
20.171.2.80 CK_OBJECT_CLASS . . . . .	1048
20.171.2.81 CK_OBJECT_CLASS_PTR . . . . .	1048
20.171.2.82 CK_OBJECT_HANDLE . . . . .	1049
20.171.2.83 CK_OBJECT_HANDLE_PTR . . . . .	1049
20.171.2.84 CK_OTP_PARAM . . . . .	1049
20.171.2.85 CK_OTP_PARAM_PTR . . . . .	1049
20.171.2.86 CK_OTP_PARAM_TYPE . . . . .	1049
20.171.2.87 CK_OTP_PARAMS . . . . .	1049
20.171.2.88 CK_OTP_PARAMS_PTR . . . . .	1049
20.171.2.89 CK_OTP_SIGNATURE_INFO . . . . .	1049
20.171.2.90 CK_OTP_SIGNATURE_INFO_PTR . . . . .	1050
20.171.2.91 CK_PARAM_TYPE . . . . .	1050
20.171.2.92 CK_PBE_PARAMS . . . . .	1050
20.171.2.93 CK_PBE_PARAMS_PTR . . . . .	1050
20.171.2.94 CK_PKCS5_PBKD2_PARAMS . . . . .	1050
20.171.2.95 CK_PKCS5_PBKD2_PARAMS2 . . . . .	1050
20.171.2.96 CK_PKCS5_PBKD2_PARAMS2_PTR . . . . .	1050
20.171.2.97 CK_PKCS5_PBKD2_PARAMS_PTR . . . . .	1050
20.171.2.98 CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE . . . . .	1051
20.171.2.99 CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE_PTR . . . . .	1051
20.171.2.100 CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE . . . . .	1051
20.171.2.101 CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE_PTR . . . . .	1051
20.171.2.102 CK_RC2_CBC_PARAMS . . . . .	1051
20.171.2.103 CK_RC2_CBC_PARAMS_PTR . . . . .	1051
20.171.2.104 CK_RC2_MAC_GENERAL_PARAMS . . . . .	1051
20.171.2.105 CK_RC2_MAC_GENERAL_PARAMS_PTR . . . . .	1051
20.171.2.106 CK_RC2_PARAMS . . . . .	1052
20.171.2.107 CK_RC2_PARAMS_PTR . . . . .	1052
20.171.2.108 CK_RC5_CBC_PARAMS . . . . .	1052
20.171.2.109 CK_RC5_CBC_PARAMS_PTR . . . . .	1052
20.171.2.110 CK_RC5_MAC_GENERAL_PARAMS . . . . .	1052
20.171.2.111 CK_RC5_MAC_GENERAL_PARAMS_PTR . . . . .	1052
20.171.2.112 CK_RC5_PARAMS . . . . .	1052
20.171.2.113 CK_RC5_PARAMS_PTR . . . . .	1052
20.171.2.114 CK_RSA_AES_KEY_WRAP_PARAMS . . . . .	1053
20.171.2.115 CK_RSA_AES_KEY_WRAP_PARAMS_PTR . . . . .	1053
20.171.2.116 CK_RSA_PKCS_MGF_TYPE . . . . .	1053
20.171.2.117 CK_RSA_PKCS_MGF_TYPE_PTR . . . . .	1053
20.171.2.118 CK_RSA_PKCS_OAEP_PARAMS . . . . .	1053
20.171.2.119 CK_RSA_PKCS_OAEP_PARAMS_PTR . . . . .	1053

20.171.2.120 CK_RSA_PKCS_OAEP_SOURCE_TYPE . . . . .	1053
20.171.2.121 CK_RSA_PKCS_OAEP_SOURCE_TYPE_PTR . . . . .	1053
20.171.2.122 CK_RSA_PKCS_PSS_PARAMS . . . . .	1054
20.171.2.123 CK_RSA_PKCS_PSS_PARAMS_PTR . . . . .	1054
20.171.2.124 CK_RV . . . . .	1054
20.171.2.125 CK_SEED_CBC_ENCRYPT_DATA_PARAMS . . . . .	1054
20.171.2.126 CK_SEED_CBC_ENCRYPT_DATA_PARAMS_PTR . . . . .	1054
20.171.2.127 CK_SESSION_HANDLE . . . . .	1054
20.171.2.128 CK_SESSION_HANDLE_PTR . . . . .	1054
20.171.2.129 CK_SESSION_INFO . . . . .	1054
20.171.2.130 CK_SESSION_INFO_PTR . . . . .	1055
20.171.2.131 CK_SKIPJACK_PRIVATE_WRAP_PARAMS . . . . .	1055
20.171.2.132 CK_SKIPJACK_PRIVATE_WRAP_PARAMS_PTR . . . . .	1055
20.171.2.133 CK_SKIPJACK_RELAYX_PARAMS . . . . .	1055
20.171.2.134 CK_SKIPJACK_RELAYX_PARAMS_PTR . . . . .	1055
20.171.2.135 CK_SLOT_ID . . . . .	1055
20.171.2.136 CK_SLOT_ID_PTR . . . . .	1055
20.171.2.137 CK_SLOT_INFO . . . . .	1055
20.171.2.138 CK_SLOT_INFO_PTR . . . . .	1056
20.171.2.139 CK_SSL3_KEY_MAT_OUT . . . . .	1056
20.171.2.140 CK_SSL3_KEY_MAT_OUT_PTR . . . . .	1056
20.171.2.141 CK_SSL3_KEY_MAT_PARAMS . . . . .	1056
20.171.2.142 CK_SSL3_KEY_MAT_PARAMS_PTR . . . . .	1056
20.171.2.143 CK_SSL3_MASTER_KEY_DERIVE_PARAMS . . . . .	1056
20.171.2.144 CK_SSL3_MASTER_KEY_DERIVE_PARAMS_PTR . . . . .	1056
20.171.2.145 CK_SSL3_RANDOM_DATA . . . . .	1056
20.171.2.146 CK_STATE . . . . .	1057
20.171.2.147 CK_TLS12_KEY_MAT_PARAMS . . . . .	1057
20.171.2.148 CK_TLS12_KEY_MAT_PARAMS_PTR . . . . .	1057
20.171.2.149 CK_TLS12_MASTER_KEY_DERIVE_PARAMS . . . . .	1057
20.171.2.150 CK_TLS12_MASTER_KEY_DERIVE_PARAMS_PTR . . . . .	1057
20.171.2.151 CK_TLS_KDF_PARAMS . . . . .	1057
20.171.2.152 CK_TLS_KDF_PARAMS_PTR . . . . .	1057
20.171.2.153 CK_TLS_MAC_PARAMS . . . . .	1057
20.171.2.154 CK_TLS_MAC_PARAMS_PTR . . . . .	1058
20.171.2.155 CK_TLS_PRF_PARAMS . . . . .	1058
20.171.2.156 CK_TLS_PRF_PARAMS_PTR . . . . .	1058
20.171.2.157 CK_TOKEN_INFO . . . . .	1058
20.171.2.158 CK_TOKEN_INFO_PTR . . . . .	1058
20.171.2.159 CK_ULONG . . . . .	1058
20.171.2.160 CK_ULONG_PTR . . . . .	1058
20.171.2.161 CK_USER_TYPE . . . . .	1058

20.171.2.162 CK_UTF8CHAR	1059
20.171.2.163 CK_UTF8CHAR_PTR	1059
20.171.2.164 CK_VERSION	1059
20.171.2.165 CK_VERSION_PTR	1059
20.171.2.166 CK_VOID_PTR	1059
20.171.2.167 CK_VOID_PTR_PTR	1059
20.171.2.168 CK_WTLS_KEY_MAT_OUT	1059
20.171.2.169 CK_WTLS_KEY_MAT_OUT_PTR	1059
20.171.2.170 CK_WTLS_KEY_MAT_PARAMS	1060
20.171.2.171 CK_WTLS_KEY_MAT_PARAMS_PTR	1060
20.171.2.172 CK_WTLS_MASTER_KEY_DERIVE_PARAMS	1060
20.171.2.173 CK_WTLS_MASTER_KEY_DERIVE_PARAMS_PTR	1060
20.171.2.174 CK_WTLS_PRF_PARAMS	1060
20.171.2.175 CK_WTLS_PRF_PARAMS_PTR	1060
20.171.2.176 CK_WTLS_RANDOM_DATA	1060
20.171.2.177 CK_WTLS_RANDOM_DATA_PTR	1060
20.171.2.178 CK_X9_42_DH1_DERIVE_PARAMS	1061
20.171.2.179 CK_X9_42_DH1_DERIVE_PARAMS_PTR	1061
20.171.2.180 CK_X9_42_DH2_DERIVE_PARAMS	1061
20.171.2.181 CK_X9_42_DH2_DERIVE_PARAMS_PTR	1061
20.171.2.182 CK_X9_42_DH_KDF_TYPE	1061
20.171.2.183 CK_X9_42_DH_KDF_TYPE_PTR	1061
20.171.2.184 CK_X9_42_MQV_DERIVE_PARAMS	1061
20.171.2.185 CK_X9_42_MQV_DERIVE_PARAMS_PTR	1061
20.171.2.186 event	1062
20.171.2.187 pApplication	1062
20.171.3 Function Documentation	1062
20.171.3.1 CK_CALLBACK_FUNCTION() [1/5]	1062
20.171.3.2 CK_CALLBACK_FUNCTION() [2/5]	1062
20.171.3.3 CK_CALLBACK_FUNCTION() [3/5]	1062
20.171.3.4 CK_CALLBACK_FUNCTION() [4/5]	1062
20.171.3.5 CK_CALLBACK_FUNCTION() [5/5]	1063
20.172 readme.md File Reference	1063
20.173 README.md File Reference	1063
20.174 README.md File Reference	1063
20.175 README.md File Reference	1063
20.176 README.md File Reference	1063
20.177 README.md File Reference	1063
20.178 README.md File Reference	1063
20.179 README.md File Reference	1063
20.180 README.md File Reference	1063
20.181 README.md File Reference	1063

---

20.182 README.md File Reference . . . . .	1063
20.183 secure_boot.c File Reference . . . . .	1063
20.183.1 Detailed Description . . . . .	1064
20.183.2 Function Documentation . . . . .	1064
20.183.2.1 bind_host_and_secure_element_with_io_protection() . . . . .	1064
20.183.2.2 secure_boot_process() . . . . .	1064
20.184 secure_boot.h File Reference . . . . .	1065
20.184.1 Detailed Description . . . . .	1065
20.184.2 Macro Definition Documentation . . . . .	1065
20.184.2.1 SECURE_BOOT_CONFIG_DISABLE . . . . .	1066
20.184.2.2 SECURE_BOOT_CONFIG_FULL_BOTH . . . . .	1066
20.184.2.3 SECURE_BOOT_CONFIG_FULL_DIG . . . . .	1066
20.184.2.4 SECURE_BOOT_CONFIG_FULL_SIGN . . . . .	1066
20.184.2.5 SECURE_BOOT_CONFIGURATION . . . . .	1066
20.184.2.6 SECURE_BOOT_DIGEST_ENCRYPT_ENABLED . . . . .	1066
20.184.2.7 SECURE_BOOT_UPGRADE_SUPPORT . . . . .	1066
20.184.3 Function Documentation . . . . .	1066
20.184.3.1 bind_host_and_secure_element_with_io_protection() . . . . .	1066
20.184.3.2 host_generate_random_number() . . . . .	1067
20.184.3.3 secure_boot_process() . . . . .	1067
20.185 secure_boot_memory.h File Reference . . . . .	1067
20.185.1 Detailed Description . . . . .	1068
20.185.2 Function Documentation . . . . .	1068
20.185.2.1 secure_boot_check_full_copy_completion() . . . . .	1068
20.185.2.2 secure_boot_deinit_memory() . . . . .	1068
20.185.2.3 secure_boot_init_memory() . . . . .	1068
20.185.2.4 secure_boot_mark_full_copy_completion() . . . . .	1068
20.185.2.5 secure_boot_read_memory() . . . . .	1068
20.185.2.6 secure_boot_write_memory() . . . . .	1069
20.186 sha1_routines.c File Reference . . . . .	1069
20.186.1 Detailed Description . . . . .	1069
20.186.2 Function Documentation . . . . .	1069
20.186.2.1 CL_hash() . . . . .	1069
20.186.2.2 CL_hashFinal() . . . . .	1070
20.186.2.3 CL_hashInit() . . . . .	1070
20.186.2.4 CL_hashUpdate() . . . . .	1070
20.186.2.5 shaEngine() . . . . .	1071
20.187 sha1_routines.h File Reference . . . . .	1071
20.187.1 Detailed Description . . . . .	1072
20.187.2 Macro Definition Documentation . . . . .	1072
20.187.2.1 _NOP . . . . .	1072
20.187.2.2 _WDRESET . . . . .	1072

20.187.2.3 leftRotate . . . . .	1072
20.187.2.4 memcpy_P . . . . .	1072
20.187.2.5 strcpy_P . . . . .	1072
20.187.2.6 U16 . . . . .	1073
20.187.2.7 U32 . . . . .	1073
20.187.2.8 U8 . . . . .	1073
20.187.3 Function Documentation . . . . .	1073
20.187.3.1 CL_hash() . . . . .	1073
20.187.3.2 CL_hashFinal() . . . . .	1073
20.187.3.3 CL_hashInit() . . . . .	1074
20.187.3.4 CL_hashUpdate() . . . . .	1074
20.187.3.5 shaEngine() . . . . .	1074
20.188 sha2_routines.c File Reference . . . . .	1074
20.188.1 Detailed Description . . . . .	1075
20.188.2 Macro Definition Documentation . . . . .	1075
20.188.2.1 rotate_right . . . . .	1075
20.188.3 Function Documentation . . . . .	1075
20.188.3.1 sw_sha256() . . . . .	1075
20.188.3.2 sw_sha256_final() . . . . .	1076
20.188.3.3 sw_sha256_init() . . . . .	1076
20.188.3.4 sw_sha256_update() . . . . .	1076
20.189 sha2_routines.h File Reference . . . . .	1077
20.189.1 Detailed Description . . . . .	1077
20.189.2 Macro Definition Documentation . . . . .	1077
20.189.2.1 SHA256_BLOCK_SIZE . . . . .	1078
20.189.2.2 SHA256_DIGEST_SIZE . . . . .	1078
20.189.3 Function Documentation . . . . .	1078
20.189.3.1 sw_sha256() . . . . .	1078
20.189.3.2 sw_sha256_final() . . . . .	1078
20.189.3.3 sw_sha256_init() . . . . .	1079
20.189.3.4 sw_sha256_update() . . . . .	1079
20.190 swi_uart_samd21_asf.c File Reference . . . . .	1079
20.190.1 Detailed Description . . . . .	1080
20.191 swi_uart_samd21_asf.h File Reference . . . . .	1080
20.191.1 Detailed Description . . . . .	1081
20.192 swi_uart_start.c File Reference . . . . .	1081
20.192.1 Detailed Description . . . . .	1082
20.192.2 Macro Definition Documentation . . . . .	1082
20.192.2.1 USART_BAUD_RATE . . . . .	1082
20.193 swi_uart_start.h File Reference . . . . .	1082
20.193.1 Detailed Description . . . . .	1083
20.194 symmetric_authentication.c File Reference . . . . .	1083



20.194.1 Detailed Description . . . . .	1084
20.194.2 Function Documentation . . . . .	1084
20.194.2.1 symmetric_authenticate() . . . . .	1084
20.195 symmetric_authentication.h File Reference . . . . .	1084
20.195.1 Detailed Description . . . . .	1085
20.195.2 Function Documentation . . . . .	1085
20.195.2.1 symmetric_authenticate() . . . . .	1085
20.196 tflxtls_cert_def_4_device.c File Reference . . . . .	1085
20.196.1 Detailed Description . . . . .	1086
20.196.2 Variable Documentation . . . . .	1086
20.196.2.1 g_tflxtls_cert_elements_4_device . . . . .	1086
20.196.2.2 g_tflxtls_cert_template_4_device . . . . .	1086
20.197 tflxtls_cert_def_4_device.h File Reference . . . . .	1086
20.197.1 Detailed Description . . . . .	1086
20.198 tng_atca.c File Reference . . . . .	1087
20.198.1 Detailed Description . . . . .	1087
20.199 tng_atca.h File Reference . . . . .	1087
20.199.1 Detailed Description . . . . .	1088
20.200 tng_atcacert_client.c File Reference . . . . .	1088
20.200.1 Detailed Description . . . . .	1089
20.200.2 Function Documentation . . . . .	1089
20.200.2.1 tng_atcacert_device_public_key() . . . . .	1089
20.200.2.2 tng_atcacert_max_signer_cert_size() . . . . .	1089
20.200.2.3 tng_atcacert_read_device_cert() . . . . .	1090
20.200.2.4 tng_atcacert_read_signer_cert() . . . . .	1090
20.200.2.5 tng_atcacert_root_cert() . . . . .	1091
20.200.2.6 tng_atcacert_root_cert_size() . . . . .	1091
20.200.2.7 tng_atcacert_root_public_key() . . . . .	1091
20.200.2.8 tng_atcacert_signer_public_key() . . . . .	1092
20.201 tng_atcacert_client.h File Reference . . . . .	1092
20.201.1 Detailed Description . . . . .	1093
20.202 tng_root_cert.c File Reference . . . . .	1093
20.202.1 Detailed Description . . . . .	1093
20.202.2 Variable Documentation . . . . .	1093
20.202.2.1 g_cryptoauth_root_ca_002_cert . . . . .	1094
20.202.2.2 g_cryptoauth_root_ca_002_cert_size . . . . .	1094
20.203 tng_root_cert.h File Reference . . . . .	1094
20.203.1 Detailed Description . . . . .	1094
20.204 tnglora_cert_def_1_signer.c File Reference . . . . .	1094
20.204.1 Detailed Description . . . . .	1095
20.204.2 Variable Documentation . . . . .	1095
20.204.2.1 g_tngtls_cert_elements_1_signer . . . . .	1095

20.204.2.2 g_tngtls_cert_template_1_signer . . . . .	1095
20.205 tnglora_cert_def_1_signer.h File Reference . . . . .	1095
20.205.1 Detailed Description . . . . .	1095
20.206 tnglora_cert_def_2_device.c File Reference . . . . .	1096
20.206.1 Detailed Description . . . . .	1096
20.206.2 Variable Documentation . . . . .	1096
20.206.2.1 g_tngtls_cert_elements_2_device . . . . .	1096
20.206.2.2 g_tngtls_cert_template_2_device . . . . .	1096
20.207 tnglora_cert_def_2_device.h File Reference . . . . .	1096
20.207.1 Detailed Description . . . . .	1097
20.208 tnglora_cert_def_4_device.c File Reference . . . . .	1097
20.208.1 Detailed Description . . . . .	1097
20.208.2 Variable Documentation . . . . .	1097
20.208.2.1 g_tnglora_cert_def_4_device . . . . .	1097
20.208.2.2 g_tnglora_cert_elements_4_device . . . . .	1098
20.208.2.3 g_tnglora_cert_template_4_device . . . . .	1098
20.209 tnglora_cert_def_4_device.h File Reference . . . . .	1098
20.209.1 Detailed Description . . . . .	1098
20.210 tngtls_cert_def_1_signer.c File Reference . . . . .	1098
20.210.1 Detailed Description . . . . .	1099
20.210.2 Variable Documentation . . . . .	1099
20.210.2.1 g_tngtls_cert_def_1_signer . . . . .	1099
20.210.2.2 g_tngtls_cert_elements_1_signer . . . . .	1099
20.210.2.3 g_tngtls_cert_template_1_signer . . . . .	1099
20.211 tngtls_cert_def_1_signer.h File Reference . . . . .	1099
20.211.1 Detailed Description . . . . .	1100
20.212 tngtls_cert_def_2_device.c File Reference . . . . .	1100
20.212.1 Detailed Description . . . . .	1100
20.212.2 Variable Documentation . . . . .	1100
20.212.2.1 g_tngtls_cert_def_2_device . . . . .	1100
20.212.2.2 g_tngtls_cert_elements_2_device . . . . .	1100
20.212.2.3 g_tngtls_cert_template_2_device . . . . .	1101
20.213 tngtls_cert_def_2_device.h File Reference . . . . .	1101
20.213.1 Detailed Description . . . . .	1101
20.214 tngtls_cert_def_3_device.c File Reference . . . . .	1101
20.214.1 Detailed Description . . . . .	1101
20.214.2 Variable Documentation . . . . .	1102
20.214.2.1 g_tngtls_cert_def_3_device . . . . .	1102
20.214.2.2 g_tngtls_cert_elements_3_device . . . . .	1102
20.214.2.3 g_tngtls_cert_template_3_device . . . . .	1102
20.215 tngtls_cert_def_3_device.h File Reference . . . . .	1102
20.215.1 Detailed Description . . . . .	1102

---

20.216 trust_pkcs11_config.c File Reference . . . . .	1102
20.216.1 Detailed Description . . . . .	1102
<b>Index</b>	<b>1103</b>

# Chapter 1

## License

MBEDTLS Interface Functions that enable mbedtls objects to use cryptoauthlib functions

Replace mbedtls ECDSA Functions with hardware acceleration & hardware key security.

Subject to your compliance with these terms, you may use Microchip software and any derivatives exclusively with Microchip products. It is your responsibility to comply with third party license terms applicable to your use of third party software (including open source software) that may accompany Microchip software.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

Replace mbedtls ECDH Functions with hardware acceleration & hardware key security.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

Subject to your compliance with these terms, you may use Microchip software and any derivatives exclusively with Microchip products. It is your responsibility to comply with third party license terms applicable to your use of third party software (including open source software) that may accompany Microchip software.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

Replace mbedtls ECDSA Functions with hardware acceleration & hardware key security

---

**Copyright**

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

Subject to your compliance with these terms, you may use Microchip software and any derivatives exclusively with Microchip products. It is your responsibility to comply with third party license terms applicable to your use of third party software (including open source software) that may accompany Microchip software.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

mbedTLS Interface Functions that enable mbedtls objects to use cryptoauthlib functions

**Copyright**

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

Subject to your compliance with these terms, you may use Microchip software and any derivatives exclusively with Microchip products. It is your responsibility to comply with third party license terms applicable to your use of third party software (including open source software) that may accompany Microchip software.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, APPLY TO THIS SOFTWARE, INCLUDING ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE SOFTWARE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

## Chapter 2

# calib directory - Purpose

The purpose of this directory is to contain the files implementing the APIs for a basic interface to the core CryptoAuthLib library.

High-level functions like these make it very convenient to use the library when standard configurations and defaults are in play. They are the easiest to use when developing examples or trying to understand the "flow" of an authentication operation without getting overwhelmed by the details.

This makes simple jobs easy and if you need more sophistication and power, you can employ the full power of the CryptoAuthLib object model.

See the Doxygen documentation in `cryptoauthlib/docs` for details on the API of the calib commands.



## **Chapter 3**

### **crypto directory - Purpose**

This directory contains software implementations of cryptographic functions. The functions at the base level are wrappers that will point to the final implementations of the software crypto functions.





## Chapter 4

# HAL Directory - Purpose

This directory contains all the Hardware Abstraction Layer (HAL) files used to adapt the upper levels of atca-ng and abstractions to physical hardware.

HAL contains physical implementations for I2C, SWI, SPI, UART and timers for specific hardware platforms.

**Include just those HAL files you require based on platform type.**

## CryptoAuthLib Supported HAL Layers

HAL Layers files are combined into groups. Initial group is generic files that are typically included in a project. Files are then broken out by uController Family and or Operating System Interface.

Protocol Files	Interface	Files	API	Notes
atca		<a href="#">atca_hal.c/h</a>		For all projects
kit protocol		<a href="#">kit_protocol.c/h</a>		For all Kit Protocol projects
		<a href="#">kit_phy.h</a>		

**Microchip Harmony 3 for all PIC32 & ARM products - Use the Harmony 3 Configurator to generate and configure projects**

Obtain library and configure using [Harmony 3](#)

Interface	Files	API	Notes
I2C	<a href="#">hal_i2c_harmony.c</a>	plib.↔ h	For all Harmony 3 based projects
SPI	<a href="#">hal_spi_harmony.c</a>	plib.↔ h	

**Microchip 8 & 16 bit products - AVR, PIC16/18, PIC24/DSPIC**

Obtain library and integration through [Microchip Code Configurator](#)

## OS & RTOS integrations

Use **CMake** to configure the library in Linux, Windows, and MacOS environments

OS	Interface	Files	API	Notes
Linux	I2C	<a href="#">hal_linux_i2c_userspace.c/h</a>	i2c-dev	
Linux	SPI	<a href="#">hal_linux_spi_userspace.c/h</a>	spidev	
Linux/Mac		<a href="#">hal_linux.c</a>		For all Linux/Mac projects
Windows		<a href="#">hal_windows.c</a>		For all Windows projects
All	kit-hid	<a href="#">hal_all_platforms_kit_hidapi.c/h</a>	hidapi	Works for Windows, Linux, and Mac
freeRTOS		<a href="#">hal_freertos.c</a>		freeRTOS common routines

## Legacy Support - **Atmel START** for AVR, ARM based processors (SAM)

Interface	Files	API	Notes
	<a href="#">hal_timer_start.c</a>	START	Timer implementation
I2C	<a href="#">hal_i2c_start.c/h</a>	START	
SWI	<a href="#">swi_uart_start.c/h</a>	START	SWI using UART

## Legacy Support - ASF3 for ARM Cortex-m0 & Cortex-m based processors (SAM)

SAM Micros	Interface	Files	API	Notes
cortex-m0	I2C	<a href="#">hal_sam0_i2c_asf.c/h</a>	ASF3	SAMD21, SAMB11, etc
cortex-m3/4/7	I2C	<a href="#">hal_sam_i2c_asf.c/h</a>	ASF3	SAM4S, SAMG55, SAMV71, etc
all		<a href="#">hal_sam_timer_asf.c</a>	ASF3	Common timer hal for all platforms

## Chapter 5

### **mbedtls directory - Purpose**

This directory contains the interfacing and wrapper functions to integrate mbedtls as the software crypto library as well as provide elliptic curve cryptography (ECC) hardware acceleration.



## **Chapter 6**

### **openssl directory - Purpose**

This directory contains the interfacing and wrapper functions to integrate openssl as the software crypto library.



## Chapter 7

# IP Protection with Symmetric Authentication

The IP protection can be easily integrated to the existing projects. The user project should include [symmetric\\_authentication.c](#) & [symmetric\\_authentication.h](#) files which contains the api

- [symmetric\\_authenticate\(\)](#) - For Performing the authentication between host & device.

## User Considerations

- The user should take care on how the master key should be stored on the MCU side.
- The api's in the file doesn't do the provisioning of the chip and user should take care of the provisioning.

With the provisioned cryptoauthentication device and after doing the cryptoauthlib initialisation, user should only be calling the function [symmetric\\_authenticate\(\)](#) with its necessary parameters for the authentication. The returned authentication status should be used in the application.

## Examples

For more information about IP protection and its example project refer [Microchip github](#)





## Chapter 8

# Setting up cryptoauthlib as a PKCS11 Provider for your system (LINUX)

These instructions are for building, installing and configuring cryptoauthlib as a pkcs11 provider. These instructions are for commonly available Linux systems with package managers.

### Update libp11 on the system. The version should be at minimum 0.4.10

- Install the build dependencies for the system:

```
```bash
```

#### Debian like systems

```
$ sudo apt-get build-dep libengine-pkcs11-openssl1.1 ```
```

```
```bash
```

#### RPM based systems

```
$ yum-builddep engine-pkcs11 ```
```

- Change to a sane directory

```
```bash cd ~ ```
```

- Get the latest version of libp11

```
```bash $ git clone https://github.com/OpenSC/libp11.git ```
```

- Rerun the build configuration tools:

```
``` $ cd libp11 $ ./bootstrap $ ./configure ```
```

- Build the library:

```
```bash $ make ```
```

- Install the library:

```
```bash $ sudo make install ```
```

## Build and Install cryptoauthlib with PKCS11 support

- Install the build dependencies for the system:

```
```bash
```

### Debian like systems

```
$ sudo apt-get install cmake libudev-dev ```
```

```
```bash
```

### RPM based systems

```
$ yum install cmake $ yum install libudev-devel ```
```

- Change to a sane directory

```
```bash cd ~ ```
```

- Get the latest version of cryptoauthlib with PKCS11 support

```
```bash $ git clone --single-branch -b pkcs11 https://github.com/MicrochipTech/cryptoauthlib
```
```

- Rerun the build configuration tools:

```
```bash $ cd cryptoauthlib $ cmake . ```
```

- Build the library:

```
```bash $ make ```
```

- Install the library:

```
```bash $ sudo make install ```
```

## Configuring the cryptoauthlib PKCS11 library

By default the following files will be created.

- /etc/cryptoauthlib/cryptoauthlib.conf

```
```text
```

### Cryptoauthlib Configuration File

```
filestore = /var/lib/cryptoauthlib ```
```

- /var/lib/cryptoauthlib/slot.conf.tmpl

```
```text
```

---

## Reserved Configuration for a device

The objects in this file will be created and marked as undeletable

These are processed in order. Configuration parameters must be comma

delimited and may not contain spaces

```
interface = i2c,0xB0 freeslots = 1,2,3
```

## Slot 0 is the primary private key

```
object = private,device,0
```

## Slot 10 is the certificate data for the device's public key

```
#object = certificate,device,10
```

## Slot 12 is the intermedate/signer certificate data

```
#object = certificate,signer,12
```

## Slot 15 is a public key

```
object = public,root,15 ``
```

## cryptoauthlib.conf

This file provides the basic configuration information for the library. The only variable is "filestore" which is where cryptoauthlib will find device specific configuration and where it will store object files from pkcs11 operations.

## slot.conf.tmpl

This is a template for device configuration files that cryptoauthlib will use to map devices and their resources into pkcs11 tokens and objects.

A device file must be named <pkcs11\_slot\_number>.conf

For a single device:

```
$ cd /var/lib/cryptoauthlib
$ cp slot.conf.tmpl 0.conf
```

Then edit 0.conf to match the device configuration being used.

**interface** Allows values: 'hid', 'i2c' If using i2c specify the address in hex for the device. This is in the device format (upper 7 bits define the address) so will not appear the same as the i2cdetect address (lower 7 bits)

**freeslots** This is a list of slots that may be used by the library when a pkcs11 operation that creates new objects is used. When the library is initialized it will scan for files of the form <pkcs11\_slot\_num>.<device\_slot\_num>.conf which defines the object using that device resource.

## Using p11-kit-proxy

This is an optional step but is very helpful for using multiple pkcs11 libraries in a system. Detailed setup can be found at [p11-glue](#)

```
# Debian like systems
$ sudo apt-get install p11-kit
# RPM based systems
$ yum install p11-kit
```

- Create or edit the global configuration file `/etc/pkcs11/pkcs11.conf`. The directory `/etc/pkcs11` may require creation first.

```

**This setting controls whether to load user configuration from the**

**`~/.config/pkcs11` directory. Possible values:**

**none: No user configuration**

**merge: Merge the user config over the system configuration (default)**

**only: Only user configuration, ignore system configuration**

`user-config: merge ````

- Create a module configuration file.

- User module name (only available for a single user): `~/.config/pkcs11/modules/cryptoauthlib.module`

- Global module name (available to the whole system): `/usr/share/p11-kit/modules/cryptoauthlib.module`  
`` module: /usr/lib/libcryptoauth.so critical: yes trust-policy: yes managed: yes log-calls: no ``

For more details on the configuration files see the [configuration documentation](#).

---

## Without using p11-kit-proxy

OpenSSL (via the libp11 project above) and p11tool support p11-kit-proxy natively so do not require additional set up if it is being used. If p11-kit-proxy is not being used then OpenSSL will have to be manually configured to use libp11 and cryptauthlib

This requires editing the default openssl.cnf file. To locate the file being used by the system run the following command:

```
$ openssl version -a | grep OPENSSLDIR:
OPENSSLDIR: "/usr/lib/ssl"
```

This gives the default path where openssl is compiled to find the openssl.cnf file

In this case the file to edit will be /usr/lib/ssl/openssl.cnf

This line must be placed at the top, before any sections are defined:

```
openssl_conf = openssl_init
```

This should be added to the bottom of the file:

```
[openssl_init]
engines=engine_section
[engine_section]
pkcs11 = pkcs11_section
[pkcs11_section]
engine_id = pkcs11
# Wherever the engine installed by libp11 is. For example it could be:
# /usr/lib/arm-linux-gnueabi/hf/engines-1.1/libpkcs11.so
dynamic_path = /usr/lib/ssl/engines/libpkcs11.so
MODULE_PATH = /usr/lib/libcryptauth.so
init = 0
```

## Testing

To use p11tool it has to be installed:

```
# Debian like systems
$ sudo apt-get install gnutls-bin
# RPM based systems
$ yum install gnutls-utils
```

**Note:** If not using p11-kit-proxy then the provider has to be specified in p11tool calls:

```
$ p11tool --provider=/usr/lib/libcryptauth.so
```

- Get the public key for a private key (as defined by the 0.conf file cited above):

```
```bash $ p11tool --export-pubkey "pkcs11:token=0123EE;object=device;type=private" warning: --login was
not specified and it may be required for this operation. warning: no --outfile was specified and the public
key will be printed on screen. -----BEGIN PUBLIC KEY----- MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQg<
AE9wzUq1EUAoNrG01rXYjNd35mxKuA Ojw/klIrNEBciSLLOTLjs/gvFS7N8AFXDK18vpxxu6yKzF2LRd7R<
Y8yEFw== -----END PUBLIC KEY----- ```
```

- Get the public key and decode it using OpenSSL

```
```bash $ p11tool --export-pubkey "pkcs11:token=0123EE;object=device;type=private" | openssl pkey -pubin
-text -noout warning: --login was not specified and it may be required for this operation. warning: no --outfile
was specified and the public key will be printed on screen. Public-Key: (256 bit) pub: 04:f7:0c:d4:ab:51<
:14:02:83:6b:1b:4d:6b:5d:88: cd:77:7e:66:c4:ab:80:3a:3c:3f:92:52:2b:34:40: 5c:89:22:cb:39:32:e3:b3:f8:2f<
:15:2e:cd:f0:01: 57:0c:ad:7c:be:9c:71:bb:ac:a4:cc:5d:8b:45:de: d1:63:cc:84:17 ASN1 OID: prime256v1 NIST
CURVE: P-256 ```
```

- Create a CSR for the private key

```
```bash $ openssl req -engine pkcs11 -key "pkcs11:token=0123EE;object=device;type=private" -keyform engine -new -out new_device.csr -subj "/CN=NEW CSR EXAMPLE" engine "pkcs11" set.
```

```
$ cat new_device.csr -----BEGIN CERTIFICATE REQUEST----- MIHVMHwCAQAwGjEYMBYGA1UEAwwPTkVXIENTUiBFWEFNUEXFMFkwEwYHkoZlZjOCAAQYIKoZlZjODAQcDQgAE9wzUq1EUAoNrG01rXYjNd35mxKuAOjw/kllrNEBciSLL OTLjs/gvFS7N8AFXDK18vpxxu6yF2LRd7RY8yEF6AAMaGcCqGSM49BAMCA0kA MEYCIQDUPeLfPcOwtZxYJDYXPdl2UhpReVn6kK2IKCCX6byM8QlhAlfqnggtcCi W21xLAzabr8A4mHyfIIQ1ofYBg8QO9jZ -----END CERTIFICATE REQUEST----- ```
```

- Verify the newly created csr

```
```bash $ openssl req -in new_device.csr -verify -text -noout verify OK Certificate Request: Data: Version: 1 (0x0) Subject: CN = NEW CSR EXAMPLE Subject Public Key Info: Public Key Algorithm: id-ecPublicKey Public-Key: (256 bit) pub: 04:f7:0c:d4:ab:51:14:02:83:6b:1b:4d:6b:5d:88: cd:77:7e:66:c4:ab:80:3a:3c:3f:92:52:2b:34:40: 5c:89:22:cb:39:32:e3:b3:f8:2f:15:2e:cd:f0:01: 57:0c:ad:7c:be:9c:71:bb:ac:a4:cc:5d:8b:45:de: d1:63:cc:84:17 ASN1 OID: prime256v1 NIST CURVE: P-256 Attributes: a0:00 Signature Algorithm: ecdsa-with-SHA256 30:46:02:21:00:d4:3d:e2:df:3d:c3:b0:b5:9c:58:24:36:17: 3d:d9:76:52:1a:51:79:59:fa:90:ad:a5:28:20:97:e9:bc:8c: f1:02:21:00:87:ea:7e:78:20:b5:c0:a2:5b:6d:71:2c:0c:da: 6e:bf:00:e2:61:f2:7c:82:10:d6:87:d8:06:0f:10:3b:d8:d9 ```
```

## Chapter 9

### app directory - Purpose

This directory is for application specific implementation of various use cases.

Methods in this directory provide a simple API to perform potentially complex combinations of calls to the main library or API.





## Chapter 10

# Secure boot using ATECC608

The SecureBoot command is a new feature on the [ATECC608A](#) device compared to earlier CryptoAuthentication devices from Microchip. This feature helps the MCU to identify fraudulent code installed on it. When this feature is implemented, the MCU can send a firmware digest and signature to the ATECC608. The ATECC608 validates this information (ECDSA verify) and responds to host with a yes or no answer.

The ATECC608 provides options to reduce the firmware verification time by storing the signature or digest after a good full verification (FullStore mode of the SecureBoot command).

- When the ATECC608 stores the digest (SecureBootMode is FullDig), the host only needs to send the firmware digest, which is compared to the stored copy. This skips the comparatively lengthy ECDSA verify, speeding up the secure boot process.
- When the ATECC608 stores the signature (SecureBootMode is FullSig), the host only needs to send the firmware digest, which is verified against the stored signature using ECDSA. This saves time by not needing to send the signature in the command over the bus.

The ATECC608 also provides wire protection features for the SecureBoot command, which can be used to encrypt the digest being sent from the host to the ATECC608 and add a MAC to the verify result coming back to the host so it can't be forced to a success state. This feature makes use of a shared secret between the host and ATECC608, called the IO protection key.

The secure boot feature can be easily integrated to an existing project. The project should include the following files from the secure\_boot folder:

- [secure\\_boot.c](#)
- [secure\\_boot.h](#)
- [secure\\_boot\\_memory.h](#)
- [io\\_protection\\_key.h](#)

The project should also implement the following platform-specific APIs:

- [secure\\_boot\\_init\\_memory\(\)](#)
- [secure\\_boot\\_read\\_memory\(\)](#)
- [secure\\_boot\\_deinit\\_memory\(\)](#)

- [secure\\_boot\\_mark\\_full\\_copy\\_completion\(\)](#)
- [secure\\_boot\\_check\\_full\\_copy\\_completion\(\)](#)
- [io\\_protection\\_get\\_key\(\)](#)
- [io\\_protection\\_set\\_key\(\)](#)

The project can set the secure boot configuration with the following defines:

- `SECURE_BOOT_CONFIGURATION`
- `SECURE_BOOT_DIGEST_ENCRYPT_ENABLED`
- `SECURE_BOOT_UPGRADE_SUPPORT`

The secure boot process is performed by initializing CryptoAuthLib and calling the [secure\\_boot\\_process\(\)](#) function.

## Implementation Considerations

- Need to perform SHA256 calculations on the host. CryptoAuthLib provides a software implementation in [lib/crypto/atca\\_crypto\\_sw\\_sha2.c](#)
- When using the wire protection features:
  - The host needs to be able to generate a nonce (number used once). This is the NumIn parameter to the Nonce command that is sent before the SecureBoot command. The ATECC608 can not be used to generate NumIn, but it should come from a good random or non-repeating source in the host.
  - If the host has any protected internal memory, it should be used to store its copy of the IO protection key.
- Secure boot depends on proper protections of the boot loader code in the host. If the code can be easily changed, then the secure boot process can be easily skipped. Boot loader should ideally be stored in an immutable (unchangeable) location like a boot ROM or write-protected flash.
- Note that these APIs don't provision the ATECC608. They assume the ATECC608 has already been configured and provisioned with the necessary keys for secure boot.

## Examples

For more information about secure boot, please see the example implementation project and documentation at: [https://github.com/MicrochipTech/cryptoauth\\_usecase\\_secureboot](https://github.com/MicrochipTech/cryptoauth_usecase_secureboot)

## Chapter 11

# TNG Functions

This folder has a number of convenience functions for working with TNG devices (currently ATECC608A-MAHTN-T).

These devices have standard certificates that can be easily read using the functions in [tng\\_atcacert\\_client.h](#)



## Chapter 12

# CryptoAuthLib - Microchip CryptoAuthentication Library

### Introduction

This library implements the APIs required to communicate with Microchip Security device. The family of devices supported currently are:

- [ATSHA204A](#)
- [ATECC108A](#)
- [ATECC508A](#)
- [ATECC608A](#)

The best place to start is with the [Microchip Trust Platform](#)

Online API documentation is at <https://microchiptech.github.io/cryptoauthlib/>

Latest software and examples can be found at:

- <https://www.microchip.com/design-centers/security-ics/trust-platform>
- <http://www.microchip.com/SWLibraryWeb/product.aspx?product=CryptoAuthLib>

Prerequisite hardware to run CryptoAuthLib examples:

- [CryptoAuth Trust Platform Development Kit](#)

Alternatively a Microchip MCU and Adapter Board:

- [ATSAMR21 Xplained Pro](#) or [ATSAMD21 Xplained Pro](#)
- [CryptoAuth Xplained Pro Extension](#) or [CryptoAuthentication SOIC Socket Board](#) to accept SOIC parts

For most development, using socketed top-boards is preferable until your configuration is well tested, then you can commit it to a CryptoAuth Xplained Pro Extension, for example. Keep in mind that once you lock a device, it will not be changeable.

## Examples

- Watch [CryptoAuthLib Documents](#) for new examples coming online.
- Node Authentication Example Using Asymmetric PKI is a complete, all-in-one example demonstrating all the stages of crypto authentication starting from provisioning the Crypto Authentication device ATECC608/ATECC508A with keys and certificates to demonstrating an authentication sequence using asymmetric techniques. <http://www.microchip.com/SWLibraryWeb/product.aspx?product=CryptoAuthLib>

## Configuration

In order to properly configured the library there must be a header file in your project named `atca_config.h` at minimum this needs to contain defines for the hal and device types being used. Most integrations have an configuration mechanism for generating this file. See the [atca\\_config.h.in](#) template which is configured by CMake for Linux, MacOS, & Windows projects.

An example of the configuration:

```
/* Cryptoauthlib Configuration File */
#ifndef ATCA_CONFIG_H
#define ATCA_CONFIG_H
/* Include HALS */
#define ATCA_HAL_I2C
/* Included device support */
#define ATCA_ATECC608_SUPPORT
/* \brief How long to wait after an initial wake failure for the POST to
 *      complete.
 * If Power-on self test (POST) is enabled, the self test will run on waking
 * from sleep or during power-on, which delays the wake reply.
 */
#ifndef ATCA_POST_DELAY_MSEC
#define ATCA_POST_DELAY_MSEC 25
#endif
#endif // ATCA_CONFIG_H
```

There are two major compiler defines that affect the operation of the library.

- `ATCA_NO_POLL` can be used to revert to a non-polling mechanism for device responses. Normally responses are polled for after sending a command, giving quicker response times. However, if `ATCA_NO_POLL` is defined, then the library will simply delay the max execution time of a command before reading the response.
- `ATCA_NO_HEAP` can be used to remove the use of malloc/free from the main library. This can be helpful for smaller MCUs that don't have a heap implemented. If just using the basic API, then there shouldn't be any code changes required. The lower-level API will no longer use the new/delete functions and the init/release functions should be used directly.

## Release notes

See Release Notes

---

## Host Device Support

CryptoAuthLib will run on a variety of platforms from small micro-controllers to desktop host systems. The current list of hardware abstraction layer support includes:

Rich OS Hosts:

- Linux Kit Protocol over HID USB
- Linux I2C
- Linux SPI
- Windows Kit Protocol over HID USB

Microcontrollers:

- Microchip AVR, SAM, & PIC families. See hal readme

If you have specific microcontrollers or Rich OS platforms you need support for, please contact us through the Microchip portal with your request.

## CryptoAuthLib Architecture

Cryptoauthlib API documentation is at <https://microchiptech.github.io/cryptoauthlib/>

The library is structured to support portability to:

- multiple hardware/microcontroller platforms
- multiple environments including bare-metal, RTOS and Windows/Linux/macOS
- multiple chip communication protocols (I2C, SPI, and SWI)

All platform dependencies are contained within the HAL (hardware abstraction layer).

## Directory Structure

```
lib - primary library source code
lib/atcacert - certificate data and i/o methods
lib/calib - the Basic Cryptoauth API
lib/crypto - Software crypto implementations external crypto libraries support (primarily SHA1 and SHA256)
lib/hal - hardware abstraction layer code for supporting specific platforms
lib/host - support functions for common host-side calculations
lib/jwt - json web token functions
test - Integration test and examples. See test/cmd-processor.c for main() implementation.
For production code, test directories should be excluded by not compiling it
into a project, so it is up to the developer to include or not as needed. Test
code adds significant bulk to an application - it's not intended to be included
in production code.
```



## Tests

There is a set of integration tests found in the test directory which will at least partially demonstrate the use of the objects. Some tests may depend upon a certain device being configured in a certain way and may not work for all devices or specific configurations of the device.

The test/cmd-processor.c file contains a main() function for running the tests. It implements a command-line interface. Typing help will bring up the list of commands available.

One first selects a device type, with one of the following commands:

- 204 (ATSHA204A)
- 108 (ATECC108A)
- 508 (ATECC508A)
- 608 (ATECC608A/B)

From there the following unit test sweets are available:

- unit (test command builder functions)
- basic (test basic API functions)
- cio (test certification i/o functions)
- cd (test certificate data functions)
- util (test utility functions)
- crypto (test software crypto functions)

Tests available depend on the lock level of the device. The unit tests won't lock the config or data zones automatically to allow retesting at desired lock levels. Therefore, some commands will need to be repeated after locking to exercise all available tests.

Starting from a blank device, the sequence of commands to exercise all unit tests is:

```
unit
basic
lockcfg
unit
basic
lockdata
unit
basic
cio
cd
util
crypto
```

## Using CryptoAuthLib (Microchip CryptoAuth Library)

The best place to start is with the [Microchip Trust Platform](#)

Also application examples are included as part of the Harmony 3 framework and can be copied from the Harmony Content Manager or found with the Harmony 3 Framework [Cryptoauthlib\\_apps](#)

---

## Incorporating CryptoAuthLib in a Linux project using USB HID devices

The Linux HID HAL files use the Linux udev development software package.

To install the udev development package under Ubuntu Linux, please type the following command at the terminal window:

```
sudo apt-get install libudev-dev
```

This adds the udev development development software package to the Ubuntu Linux installation.

The Linux HID HAL files also require a udev rule to be added to change the permissions of the USB HID Devices. Please add a new udev rule for the Microchip CryptoAuth USB devices.

```
cd /etc/udev/rules.d
sudo touch mchp-cryptoauth.rules
```

Edit the mchp-cryptoauth.rules file and add the following line to the file:

```
SUBSYSTEM=="hidraw", ATTRS{idVendor}=="03eb", ATTRS{idProduct}=="2312", MODE="0666"
```



## Chapter 13

# Deprecated List

### Global `atcab_init_device` (ATCADevice ca\_device)

This function is not recommended for use generally. Use of `_ext` is recommended instead. You can use `atcab↵_init_ext` to obtain an initialized instance and associated it with the global structure - but this shouldn't be a required process except in extremely unusual circumstances.



## Chapter 14

## Todo List

Global `pkcs11_init` (`CK_C_INITIALIZE_ARGS_PTR pInitArgs`)

This is where we should allocate a new context if we're using dynamic memory

If we're using dynamic memory we need to make sure to deallocate it if any of the errors after the allocations are encountered

Global `pkcs11_deinit` (`CK_VOID_PTR pReserved`)

If other threads are waiting for something to happen this call should cause those calls to unblock and return `CKR_CRYPTOKI_NOT_INITIALIZED` - How that is done by this simplified mutex API is yet to be determined



# Chapter 15

## Module Index

### 15.1 Modules

Here is a list of all modules:

Basic Crypto API methods (atcab_)	51
Configuration (cfg_)	121
ATCACCommand (atca_)	122
ATCADevice (atca_)	124
ATCAIface (atca_)	146
Certificate manipulation methods (atcacert_)	153
Basic Crypto API methods for CryptoAuth Devices (calib_)	201
Software crypto methods (atcac_)	251
Hardware abstraction layer (hal_)	257
Host side crypto methods (atcah_)	308
JSON Web Token (JWT) methods (atca_jwt_)	331
mbedTLS Wrapper methods (atca_mbedtls_)	333
Attributes (pkcs11_attr_)	335
TNG API (tng_)	378





## Chapter 16

# Data Structure Index

### 16.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">_atecc508a_config</a>	387
<a href="#">_atecc608_config</a>	390
<a href="#">_atsha204a_config</a>	395
<a href="#">_pkcs11_mech_table_e</a>	398
<a href="#">_pkcs11_attr_model</a>	398
<a href="#">_pkcs11_lib_ctx</a>	399
<a href="#">_pkcs11_object</a>	400
<a href="#">_pkcs11_object_cache_t</a>	402
<a href="#">_pkcs11_session_ctx</a>	403
<a href="#">_pkcs11_slot_ctx</a>	405
<a href="#">atca_aes_cbc_ctx</a>	407
<a href="#">atca_aes_cmac_ctx</a>	408
<a href="#">atca_aes_ctr_ctx</a>	409
<a href="#">atca_aes_gcm_ctx</a>	410
<a href="#">atca_check_mac_in_out</a>	
Input/output parameters for function <a href="#">atcah_check_mac()</a>	413
<a href="#">atca_command</a>	
Atca_command is the C object backing ATCACCommand	415
<a href="#">atca_decrypt_in_out</a>	
Input/output parameters for function <a href="#">atca_decrypt()</a>	416
<a href="#">atca_derive_key_in_out</a>	
Input/output parameters for function <a href="#">atcah_derive_key()</a>	417
<a href="#">atca_derive_key_mac_in_out</a>	
Input/output parameters for function <a href="#">atcah_derive_key_mac()</a>	418
<a href="#">atca_device</a>	
Atca_device is the C object backing ATCADevice. See the <a href="#">atca_device.h</a> file for details on the ATCADevice methods	420
<a href="#">atca_gen_dig_in_out</a>	
Input/output parameters for function <a href="#">atcah_gen_dig()</a>	421
<a href="#">atca_gen_key_in_out</a>	
Input/output parameters for calculating the PubKey digest put into TempKey by the GenKey command with the <a href="#">atcah_gen_key_msg()</a> function	424
<a href="#">atca_hmac_in_out</a>	
Input/output parameters for function <a href="#">atca_hmac()</a>	425

<a href="#">atca_iface</a>	
Atca_iface is the C object backing ATCAIface. See the <a href="#">atca_iface.h</a> file for details on the ATCAIface methods	426
<a href="#">atca_include_data_in_out</a>	
Input / output parameters for function <a href="#">atca_include_data()</a>	428
<a href="#">atca_io_decrypt_in_out</a>	428
<a href="#">atca_jwt_t</a>	
Structure to hold metadata information about the jwt being built	430
<a href="#">atca_mac_in_out</a>	
Input/output parameters for function <a href="#">atca_mac()</a>	430
<a href="#">atca_nonce_in_out</a>	
Input/output parameters for function <a href="#">atca_nonce()</a>	431
<a href="#">atca_secureboot_enc_in_out</a>	432
<a href="#">atca_secureboot_mac_in_out</a>	433
<a href="#">atca_sha256_ctx</a>	435
<a href="#">atca_sign_internal_in_out</a>	
Input/output parameters for calculating the message and digest used by the Sign(internal) command. Used with the <a href="#">atcah_sign_internal_msg()</a> function	435
<a href="#">atca_temp_key</a>	
Structure to hold TempKey fields	439
<a href="#">atca_verify_in_out</a>	
Input/output parameters for function <a href="#">atcah_verify()</a>	441
<a href="#">atca_verify_mac</a>	441
<a href="#">atca_write_mac_in_out</a>	
Input/output parameters for function <a href="#">atcah_write_auth_mac()</a> and <a href="#">atcah_privwrite_auth_mac()</a>	443
<a href="#">atcacert_build_state_s</a>	445
<a href="#">atcacert_cert_element_s</a>	447
<a href="#">atcacert_cert_loc_s</a>	448
<a href="#">atcacert_def_s</a>	448
<a href="#">atcacert_device_loc_s</a>	453
<a href="#">atcacert_tm_utc_s</a>	454
<a href="#">ATCAHAL_t</a>	
Intermediary data structure to allow the HAL layer to point the standard API functions used by the upper layers to the HAL implementation for the interface. This isolates the upper layers and loosely couples the ATCAIface object from the physical implementation	455
<a href="#">atcahid</a>	457
<a href="#">atcal2Cmaster</a>	
This is the hal_data for ATCA HAL for ASF SERCOM	458
<a href="#">ATCAIfaceCfg</a>	459
<a href="#">ATCAPacket</a>	464
<a href="#">atcaSPImaster</a>	465
<a href="#">atcaSWImaster</a>	
This is the hal_data for ATCA HAL for ASF SERCOM	466
<a href="#">CK_AES_CBC_ENCRYPT_DATA_PARAMS</a>	467
<a href="#">CK_AES_CCM_PARAMS</a>	468
<a href="#">CK_AES_CTR_PARAMS</a>	469
<a href="#">CK_AES_GCM_PARAMS</a>	470
<a href="#">CK_ARIA_CBC_ENCRYPT_DATA_PARAMS</a>	471
<a href="#">CK_ATTRIBUTE</a>	471
<a href="#">CK_C_INITIALIZE_ARGS</a>	472
<a href="#">CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS</a>	473
<a href="#">CK_CAMELLIA_CTR_PARAMS</a>	474
<a href="#">CK_CCM_PARAMS</a>	475
<a href="#">CK_CMS_SIG_PARAMS</a>	476
<a href="#">CK_DATE</a>	477
<a href="#">CK_DES_CBC_ENCRYPT_DATA_PARAMS</a>	478
<a href="#">CK_DSA_PARAMETER_GEN_PARAM</a>	478
<a href="#">CK_ECDH1_DERIVE_PARAMS</a>	479

CK_ECDH2_DERIVE_PARAMS	480
CK_ECDH_AES_KEY_WRAP_PARAMS	482
CK_ECMQV_DERIVE_PARAMS	483
CK_FUNCTION_LIST	484
CK_GCM_PARAMS	485
CK_GOSTR3410_DERIVE_PARAMS	486
CK_GOSTR3410_KEY_WRAP_PARAMS	487
CK_INFO	488
CK_KEA_DERIVE_PARAMS	489
CK_KEY_DERIVATION_STRING_DATA	490
CK_KEY_WRAP_SET_OAEP_PARAMS	491
CK_KIP_PARAMS	491
CK_MECHANISM	492
CK_MECHANISM_INFO	493
CK_OTP_PARAM	494
CK_OTP_PARAMS	494
CK_OTP_SIGNATURE_INFO	495
CK_PBE_PARAMS	496
CK_PKCS5_PBKD2_PARAMS	497
CK_PKCS5_PBKD2_PARAMS2	498
CK_RC2_CBC_PARAMS	500
CK_RC2_MAC_GENERAL_PARAMS	500
CK_RC5_CBC_PARAMS	501
CK_RC5_MAC_GENERAL_PARAMS	502
CK_RC5_PARAMS	502
CK_RSA_AES_KEY_WRAP_PARAMS	503
CK_RSA_PKCS_OAEP_PARAMS	504
CK_RSA_PKCS_PSS_PARAMS	505
CK_SEED_CBC_ENCRYPT_DATA_PARAMS	505
CK_SESSION_INFO	506
CK_SKIPJACK_PRIVATE_WRAP_PARAMS	507
CK_SKIPJACK_RELAYX_PARAMS	509
CK_SLOT_INFO	511
CK_SSL3_KEY_MAT_OUT	512
CK_SSL3_KEY_MAT_PARAMS	513
CK_SSL3_MASTER_KEY_DERIVE_PARAMS	514
CK_SSL3_RANDOM_DATA	515
CK_TLS12_KEY_MAT_PARAMS	516
CK_TLS12_MASTER_KEY_DERIVE_PARAMS	517
CK_TLS_KDF_PARAMS	518
CK_TLS_MAC_PARAMS	519
CK_TLS_PRF_PARAMS	519
CK_TOKEN_INFO	521
CK_VERSION	523
CK_WTLS_KEY_MAT_OUT	524
CK_WTLS_KEY_MAT_PARAMS	525
CK_WTLS_MASTER_KEY_DERIVE_PARAMS	526
CK_WTLS_PRF_PARAMS	527
CK_WTLS_RANDOM_DATA	528
CK_X9_42_DH1_DERIVE_PARAMS	529
CK_X9_42_DH2_DERIVE_PARAMS	530
CK_X9_42_MQV_DERIVE_PARAMS	532
CL_HashContext	533
hid_device	534
hw_sha256_ctx	535
i2c_sam0_instance	536
i2c_sam_instance	536
i2c_start_instance	537

<a href="#">memory_parameters</a>	538
<a href="#">secure_boot_config_bits</a>	539
<a href="#">secure_boot_parameters</a>	540
<a href="#">sw_sha256_ctx</a>	540
<a href="#">tng_cert_map_element</a>	542

## Chapter 17

# File Index

### 17.1 File List

Here is a list of all files with brief descriptions:

<a href="#">api_206a.c</a>	Provides APIs to use with ATSHA206A device . . . . .	543
<a href="#">api_206a.h</a>	Provides api interfaces to use with ATSHA206A device . . . . .	549
<a href="#">atca_basic.c</a>	CryptoAuthLib Basic API methods. These methods provide a simpler way to access the core crypto methods . . . . .	557
<a href="#">atca_basic.h</a>	CryptoAuthLib Basic API methods - a simple crypto authentication API. These methods manage a global ATCADevice object behind the scenes. They also manage the wake/idle state transitions so callers don't need to . . . . .	563
<a href="#">atca_bool.h</a>	Bool define for systems that don't have it . . . . .	571
<a href="#">atca_cfgs.c</a>	Set of default configurations for various ATCA devices and interfaces . . . . .	571
<a href="#">atca_cfgs.h</a>	Set of default configurations for various ATCA devices and interfaces . . . . .	571
<a href="#">atca_command.c</a>	Microchip CryptoAuthentication device command builder - this is the main object that builds the command byte strings for the given device. It does not execute the command. The basic flow is to call a command method to build the command you want given the parameters and then send that byte string through the device interface . . . . .	574
<a href="#">atca_command.h</a>	Microchip Crypto Auth device command object - this is a command builder only, it does not send the command. The result of a command method is a fully formed packet, ready to send to the ATCAIFace object to dispatch . . . . .	574
<a href="#">atca_compiler.h</a>	CryptoAuthLib is meant to be portable across architectures, even non-Microchip architectures and compiler environments. This file is for isolating compiler specific macros . . . . .	575
<a href="#">atca_crypto_hw_aes.h</a>	AES CTR, CBC & CMAC structure definitions . . . . .	575
<a href="#">atca_crypto_hw_aes_cbc.c</a>	CryptoAuthLib Basic API methods for AES CBC mode . . . . .	576
<a href="#">atca_crypto_hw_aes_cmac.c</a>	CryptoAuthLib Basic API methods for AES CBC_MAC mode . . . . .	577

<a href="#">atca_crypto_hw_aes_ctr.c</a>	
CryptoAuthLib Basic API methods for AES CTR mode	578
<a href="#">atca_crypto_sw.h</a>	
Common defines for CryptoAuthLib software crypto wrappers	579
<a href="#">atca_crypto_sw_ecdsa.c</a>	
API wrapper for software ECDSA verify. Currently unimplemented but could be implemented via a 3rd party library such as MicroECC	585
<a href="#">atca_crypto_sw_ecdsa.h</a>	585
<a href="#">atca_crypto_sw_rand.c</a>	
API wrapper for software random	586
<a href="#">atca_crypto_sw_rand.h</a>	586
<a href="#">atca_crypto_sw_sha1.c</a>	
Wrapper API for SHA 1 routines	587
<a href="#">atca_crypto_sw_sha1.h</a>	
Wrapper API for SHA 1 routines	587
<a href="#">atca_crypto_sw_sha2.c</a>	
Wrapper API for software SHA 256 routines	588
<a href="#">atca_crypto_sw_sha2.h</a>	
Wrapper API for software SHA 256 routines	588
<a href="#">atca_debug.c</a>	
Debug/Trace for CryptoAuthLib calls	589
<a href="#">atca_debug.h</a>	590
<a href="#">atca_device.c</a>	
Microchip CryptoAuth device object	591
<a href="#">atca_device.h</a>	
Microchip Crypto Auth device object	592
<a href="#">atca_devtypes.h</a>	
Microchip Crypto Auth	595
<a href="#">atca_hal.c</a>	
Low-level HAL - methods used to setup indirection to physical layer interface. this level does the dirty work of abstracting the higher level ATCAIFace methods from the low-level physical interfaces. Its main goal is to keep low-level details from bleeding into the logical interface implementation	596
<a href="#">atca_hal.h</a>	
Low-level HAL - methods used to setup indirection to physical layer interface	596
<a href="#">atca_helpers.c</a>	
Helpers to support the CryptoAuthLib Basic API methods	597
<a href="#">atca_helpers.h</a>	
Helpers to support the CryptoAuthLib Basic API methods	609
<a href="#">atca_host.c</a>	
Host side methods to support CryptoAuth computations	620
<a href="#">atca_host.h</a>	
Definitions and Prototypes for ATCA Utility Functions	621
<a href="#">atca_iface.c</a>	
Microchip CryptoAuthLib hardware interface object	624
<a href="#">atca_iface.h</a>	
Microchip Crypto Auth hardware interface object	625
<a href="#">atca_jwt.c</a>	
Utilities to create and verify a JSON Web Token (JWT)	627
<a href="#">atca_jwt.h</a>	
Utilities to create and verify a JSON Web Token (JWT)	627
<a href="#">atca_mbedtls_ecdh.c</a>	628
<a href="#">atca_mbedtls_ecdsa.c</a>	628
<a href="#">atca_mbedtls_wrap.c</a>	
Wrapper functions to replace cryptoauthlib software crypto functions with the mbedtls equivalent	628
<a href="#">atca_mbedtls_wrap.h</a>	636

<a href="#">atca_openssl_interface.c</a>	
Crypto abstraction functions for external host side cryptography . . . . .	636
<a href="#">atca_start_config.h</a> . . . . .	643
<a href="#">atca_start_iface.h</a> . . . . .	643
<a href="#">atca_status.h</a>	
Microchip Crypto Auth status codes . . . . .	643
<a href="#">atca_version.h</a>	
Microchip CryptoAuth Library Version . . . . .	646
<a href="#">atca_wolfssl_interface.c</a>	
Crypto abstraction functions for external host side cryptography . . . . .	647
<a href="#">atcacert.h</a>	
Declarations common to all atcacert code . . . . .	647
<a href="#">atcacert_client.c</a>	
Client side cert i/o methods. These declarations deal with the client-side, the node being authenticated, of the authentication process. It is assumed the client has an ECC CryptoAuthentication device (e.g. ATECC508A) and the certificates are stored on that device . . . . .	648
<a href="#">atcacert_client.h</a>	
Client side cert i/o methods. These declarations deal with the client-side, the node being authenticated, of the authentication process. It is assumed the client has an ECC CryptoAuthentication device (e.g. ATECC508A) and the certificates are stored on that device . . . . .	649
<a href="#">atcacert_date.c</a>	
Date handling with regard to certificates . . . . .	650
<a href="#">atcacert_date.h</a>	
Declarations for date handling with regard to certificates . . . . .	651
<a href="#">atcacert_def.c</a>	
Main certificate definition implementation . . . . .	653
<a href="#">atcacert_def.h</a>	
Declarations for certificates related to ECC CryptoAuthentication devices. These are the definitions required to define a certificate and its various elements with regards to the CryptoAuthentication ECC devices . . . . .	656
<a href="#">atcacert_der.c</a>	
Functions required to work with DER encoded data related to X.509 certificates . . . . .	660
<a href="#">atcacert_der.h</a>	
Function declarations required to work with DER encoded data related to X.509 certificates . . . . .	660
<a href="#">atcacert_host_hw.c</a>	
Host side methods using CryptoAuth hardware . . . . .	661
<a href="#">atcacert_host_hw.h</a>	
Host side methods using CryptoAuth hardware . . . . .	662
<a href="#">atcacert_host_sw.c</a>	
Host side methods using software implementations . . . . .	662
<a href="#">atcacert_host_sw.h</a>	
Host side methods using software implementations. host-side, the one authenticating a client, of the authentication process. Crypto functions are performed using a software library . . . . .	663
<a href="#">atcacert_pem.c</a>	
Functions required to work with PEM encoded data related to X.509 certificates . . . . .	664
<a href="#">atcacert_pem.h</a>	
Functions for converting between DER and PEM formats . . . . .	667
<a href="#">calib_aes.c</a>	
CryptoAuthLib Basic API methods for AES command . . . . .	672
<a href="#">calib_aes_gcm.c</a>	
CryptoAuthLib Basic API methods for AES GCM mode . . . . .	673
<a href="#">calib_aes_gcm.h</a>	
Unity tests for the cryptoauthlib AES GCM functions . . . . .	677
<a href="#">calib_basic.c</a>	
CryptoAuthLib Basic API methods. These methods provide a simpler way to access the core crypto methods . . . . .	678
<a href="#">calib_basic.h</a> . . . . .	679



<a href="#">calib_checkmac.c</a>	CryptoAuthLib Basic API methods for CheckMAC command . . . . .	685
<a href="#">calib_command.c</a>	Microchip CryptoAuthentication device command builder - this is the main object that builds the command byte strings for the given device. It does not execute the command. The basic flow is to call a command method to build the command you want given the parameters and then send that byte string through the device interface . . . . .	685
<a href="#">calib_command.h</a>	Microchip Crypto Auth device command object - this is a command builder only, it does not send the command. The result of a command method is a fully formed packet, ready to send to the ATCAIFace object to dispatch . . . . .	698
<a href="#">calib_counter.c</a>	CryptoAuthLib Basic API methods for Counter command . . . . .	801
<a href="#">calib_derivekey.c</a>	CryptoAuthLib Basic API methods for DeriveKey command . . . . .	801
<a href="#">calib_ecdh.c</a>	CryptoAuthLib Basic API methods for ECDH command . . . . .	802
<a href="#">calib_execution.c</a>	Implements an execution handler that executes a given command on a device and returns the results . . . . .	804
<a href="#">calib_execution.h</a>	Defines an execution handler that executes a given command on a device and returns the results	805
<a href="#">calib_gendig.c</a>	CryptoAuthLib Basic API methods for GenDig command . . . . .	806
<a href="#">calib_genkey.c</a>	CryptoAuthLib Basic API methods for GenKey command . . . . .	807
<a href="#">calib_hmac.c</a>	CryptoAuthLib Basic API methods for HMAC command . . . . .	807
<a href="#">calib_info.c</a>	CryptoAuthLib Basic API methods for Info command . . . . .	808
<a href="#">calib_kdf.c</a>	CryptoAuthLib Basic API methods for KDF command . . . . .	809
<a href="#">calib_lock.c</a>	CryptoAuthLib Basic API methods for Lock command . . . . .	809
<a href="#">calib_mac.c</a>	CryptoAuthLib Basic API methods for MAC command . . . . .	810
<a href="#">calib_nonce.c</a>	CryptoAuthLib Basic API methods for Nonce command . . . . .	811
<a href="#">calib_privwrite.c</a>	CryptoAuthLib Basic API methods for PrivWrite command . . . . .	811
<a href="#">calib_random.c</a>	CryptoAuthLib Basic API methods for Random command . . . . .	813
<a href="#">calib_read.c</a>	CryptoAuthLib Basic API methods for Read command . . . . .	813
<a href="#">calib_secureboot.c</a>	CryptoAuthLib Basic API methods for SecureBoot command . . . . .	815
<a href="#">calib_selftest.c</a>	CryptoAuthLib Basic API methods for SelfTest command . . . . .	816
<a href="#">calib_sha.c</a>	CryptoAuthLib Basic API methods for SHA command . . . . .	816
<a href="#">calib_sign.c</a>	CryptoAuthLib Basic API methods for Sign command . . . . .	818
<a href="#">calib_updateextra.c</a>	CryptoAuthLib Basic API methods for UpdateExtra command . . . . .	819
<a href="#">calib_verify.c</a>	CryptoAuthLib Basic API methods for Verify command . . . . .	819
<a href="#">calib_write.c</a>	CryptoAuthLib Basic API methods for Write command . . . . .	820

<a href="#">cryptoauthlib.h</a>	Single aggregation point for all CryptoAuthLib header files	822
<a href="#">cryptoki.h</a>		826
<a href="#">example_cert_chain.c</a>		828
<a href="#">example_cert_chain.h</a>		830
<a href="#">example_pkcs11_config.c</a>		831
<a href="#">hal_all_platforms_kit_hidapi.c</a>	HAL for kit protocol over HID for any platform	833
<a href="#">hal_all_platforms_kit_hidapi.h</a>	HAL for kit protocol over HID for any platform	834
<a href="#">hal_esp32_i2c.c</a>		835
<a href="#">hal_esp32_timer.c</a>		847
<a href="#">hal_freertos.c</a>	FreeRTOS Hardware/OS Abstraction Layer	848
<a href="#">hal_i2c_harmony.c</a>	ATCA Hardware abstraction layer for SAMD21 I2C over Harmony PLIB	848
<a href="#">hal_i2c_start.c</a>	ATCA Hardware abstraction layer for SAMD21 I2C over START drivers	849
<a href="#">hal_i2c_start.h</a>	ATCA Hardware abstraction layer for SAMD21 I2C over START drivers	850
<a href="#">hal_linux.c</a>	Timer Utility Functions for Linux	851
<a href="#">hal_linux_i2c_userspace.c</a>	ATCA Hardware abstraction layer for Linux using I2C	852
<a href="#">hal_linux_i2c_userspace.h</a>	ATCA Hardware abstraction layer for Linux using I2C	853
<a href="#">hal_linux_kit_hid.c</a>	ATCA Hardware abstraction layer for Linux using kit protocol over a USB HID device	853
<a href="#">hal_linux_kit_hid.h</a>	ATCA Hardware abstraction layer for Linux using kit protocol over a USB HID device	854
<a href="#">hal_linux_spi_userspace.c</a>		855
<a href="#">hal_linux_spi_userspace.h</a>	ATCA Hardware abstraction layer for Linux using SPI	860
<a href="#">hal_sam0_i2c_asf.c</a>	ATCA Hardware abstraction layer for SAMD21 I2C over ASF drivers	860
<a href="#">hal_sam0_i2c_asf.h</a>	ATCA Hardware abstraction layer for SAMD21 I2C over ASF drivers	861
<a href="#">hal_sam_i2c_asf.c</a>	ATCA Hardware abstraction layer for SAM flexcom & twi I2C over ASF drivers	862
<a href="#">hal_sam_i2c_asf.h</a>	ATCA Hardware abstraction layer for SAMG55 I2C over ASF drivers	863
<a href="#">hal_sam_timer_asf.c</a>	ATCA Hardware abstraction layer for SAMD21 timer/delay over ASF drivers	864
<a href="#">hal_spi_harmony.c</a>	ATCA Hardware abstraction layer for SPI over Harmony PLIB	865
<a href="#">hal_swi_uart.c</a>	ATCA Hardware abstraction layer for SWI over UART drivers	866
<a href="#">hal_swi_uart.h</a>	ATCA Hardware abstraction layer for SWI over UART drivers	867
<a href="#">hal_timer_start.c</a>	ATCA Hardware abstraction layer for SAMD21 I2C over START drivers	867
<a href="#">hal_uc3_i2c_asf.c</a>	ATCA Hardware abstraction layer for SAMV71 I2C over ASF drivers	868
<a href="#">hal_uc3_i2c_asf.h</a>	ATCA Hardware abstraction layer for SAMV71 I2C over ASF drivers	869
<a href="#">hal_uc3_timer_asf.c</a>	ATCA Hardware abstraction layer for SAM4S I2C over ASF drivers	870

<a href="#">hal_win_kit_hid.c</a>	ATCA Hardware abstraction layer for Windows using kit protocol over a USB HID device . . . .	870
<a href="#">hal_win_kit_hid.h</a>	ATCA Hardware abstraction layer for Windows using kit protocol over a USB HID device . . . .	871
<a href="#">hal_windows.c</a>	ATCA Hardware abstraction layer for windows timer functions . . . . .	872
<a href="#">io_protection_key.h</a>	Provides required interface to access IO protection key . . . . .	873
<a href="#">kit_phy.h</a>	ATCA Hardware abstraction layer physical send & receive function definitions . . . . .	874
<a href="#">kit_protocol.c</a>	Microchip Crypto Auth hardware interface object . . . . .	874
<a href="#">kit_protocol.h</a>		875
<a href="#">pkcs11.h</a>		894
<a href="#">pkcs11_attrib.c</a>	PKCS11 Library Object Attributes Handling . . . . .	895
<a href="#">pkcs11_attrib.h</a>	PKCS11 Library Object Attribute Handling . . . . .	896
<a href="#">pkcs11_cert.c</a>	PKCS11 Library Certificate Handling . . . . .	897
<a href="#">pkcs11_cert.h</a>	PKCS11 Library Certificate Handling . . . . .	898
<a href="#">pkcs11_config.c</a>	PKCS11 Library Configuration . . . . .	899
<a href="#">pkcs11_debug.c</a>	PKCS11 Library Debugging . . . . .	900
<a href="#">pkcs11_debug.h</a>	PKCS11 Library Debugging . . . . .	900
<a href="#">pkcs11_digest.c</a>		901
<a href="#">pkcs11_digest.h</a>	PKCS11 Library Digest (SHA256) Handling . . . . .	903
<a href="#">pkcs11_find.c</a>	PKCS11 Library Object Find/Searching . . . . .	904
<a href="#">pkcs11_find.h</a>	PKCS11 Library Object Find/Searching . . . . .	905
<a href="#">pkcs11_info.c</a>	PKCS11 Library Information Functions . . . . .	905
<a href="#">pkcs11_info.h</a>	PKCS11 Library Information Functions . . . . .	906
<a href="#">pkcs11_init.c</a>	PKCS11 Library Init/Deinit . . . . .	907
<a href="#">pkcs11_init.h</a>	PKCS11 Library Initialization & Context . . . . .	907
<a href="#">pkcs11_key.c</a>	PKCS11 Library Key Object Handling . . . . .	909
<a href="#">pkcs11_key.h</a>	PKCS11 Library Object Handling . . . . .	910
<a href="#">pkcs11_main.c</a>	PKCS11 Basic library redirects based on the 2.40 specification <a href="http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.html">http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.html</a> . . . . .	910
<a href="#">pkcs11_mech.c</a>	PKCS11 Library Mechanism Handling . . . . .	914
<a href="#">pkcs11_mech.h</a>	PKCS11 Library Mechanism Handling . . . . .	915
<a href="#">pkcs11_object.c</a>	PKCS11 Library Object Handling Base . . . . .	916

<a href="#">pkcs11_object.h</a>	PKCS11 Library Object Handling . . . . .	917
<a href="#">pkcs11_os.c</a>	PKCS11 Library Operating System Abstraction Functions . . . . .	920
<a href="#">pkcs11_os.h</a>	PKCS11 Library Operating System Abstraction . . . . .	920
<a href="#">pkcs11_session.c</a>	PKCS11 Library Session Handling . . . . .	921
<a href="#">pkcs11_session.h</a>	PKCS11 Library Session Management & Context . . . . .	922
<a href="#">pkcs11_signature.c</a>	PKCS11 Library Sign/Verify Handling . . . . .	923
<a href="#">pkcs11_signature.h</a>	PKCS11 Library Sign/Verify Handling . . . . .	924
<a href="#">pkcs11_slot.c</a>	PKCS11 Library Slot Handling . . . . .	925
<a href="#">pkcs11_slot.h</a>	PKCS11 Library Slot Handling & Context . . . . .	926
<a href="#">pkcs11_token.c</a>	PKCS11 Library Token Handling . . . . .	927
<a href="#">pkcs11_token.h</a>	PKCS11 Library Token Management & Context . . . . .	928
<a href="#">pkcs11_util.c</a>	PKCS11 Library Utility Functions . . . . .	929
<a href="#">pkcs11_util.h</a>	PKCS11 Library Utilities . . . . .	929
<a href="#">pkcs11f.h</a>		930
<a href="#">pkcs11t.h</a>		930
<a href="#">secure_boot.c</a>	Provides required APIs to manage secure boot under various scenarios . . . . .	1063
<a href="#">secure_boot.h</a>	Provides required APIs to manage secure boot under various scenarios . . . . .	1065
<a href="#">secure_boot_memory.h</a>	Provides interface to memory component for the secure boot . . . . .	1067
<a href="#">sha1_routines.c</a>	Software implementation of the SHA1 algorithm . . . . .	1069
<a href="#">sha1_routines.h</a>	Software implementation of the SHA1 algorithm . . . . .	1071
<a href="#">sha2_routines.c</a>	Software implementation of the SHA256 algorithm . . . . .	1074
<a href="#">sha2_routines.h</a>	Software implementation of the SHA256 algorithm . . . . .	1077
<a href="#">swi_uart_samd21_asf.c</a>	ATXMEGA's ATCA Hardware abstraction layer for SWI interface over UART drivers . . . . .	1079
<a href="#">swi_uart_samd21_asf.h</a>	ATXMEGA's ATCA Hardware abstraction layer for SWI interface over UART drivers . . . . .	1080
<a href="#">swi_uart_start.c</a>		1081
<a href="#">swi_uart_start.h</a>		1082
<a href="#">symmetric_authentication.c</a>	Contains API for performing the symmetric Authentication between the Host and the device . . . . .	1083
<a href="#">symmetric_authentication.h</a>	Contains API for performing the symmetric Authentication between the Host and the device . . . . .	1084
<a href="#">tflxtls_cert_def_4_device.c</a>	TNG TLS device certificate definition . . . . .	1085
<a href="#">tflxtls_cert_def_4_device.h</a>	TNG TLS device certificate definition . . . . .	1086
<a href="#">tng_atca.c</a>	TNG Helper Functions . . . . .	1087

<a href="#">tng_atca.h</a>	
TNG Helper Functions . . . . .	1087
<a href="#">tng_atcacert_client.c</a>	
Client side certificate I/O functions for TNG devices . . . . .	1088
<a href="#">tng_atcacert_client.h</a>	
Client side certificate I/O functions for TNG devices . . . . .	1092
<a href="#">tng_root_cert.c</a>	
TNG root certificate (DER) . . . . .	1093
<a href="#">tng_root_cert.h</a>	
TNG root certificate (DER) . . . . .	1094
<a href="#">tnglora_cert_def_1_signer.c</a>	
TNG LORA signer certificate definition . . . . .	1094
<a href="#">tnglora_cert_def_1_signer.h</a>	
TNG LORA signer certificate definition . . . . .	1095
<a href="#">tnglora_cert_def_2_device.c</a>	
TNG LORA device certificate definition . . . . .	1096
<a href="#">tnglora_cert_def_2_device.h</a>	
TNG LORA device certificate definition . . . . .	1096
<a href="#">tnglora_cert_def_4_device.c</a>	
TNG LORA device certificate definition . . . . .	1097
<a href="#">tnglora_cert_def_4_device.h</a>	
TNG LORA device certificate definition . . . . .	1098
<a href="#">tngtls_cert_def_1_signer.c</a>	
TNG TLS signer certificate definition . . . . .	1098
<a href="#">tngtls_cert_def_1_signer.h</a>	
TNG TLS signer certificate definition . . . . .	1099
<a href="#">tngtls_cert_def_2_device.c</a>	
TNG TLS device certificate definition . . . . .	1100
<a href="#">tngtls_cert_def_2_device.h</a>	
TNG TLS device certificate definition . . . . .	1101
<a href="#">tngtls_cert_def_3_device.c</a>	
TNG TLS device certificate definition . . . . .	1101
<a href="#">tngtls_cert_def_3_device.h</a>	
TNG TLS device certificate definition . . . . .	1102
<a href="#">trust_pkcs11_config.c</a>	
PKCS11 Trust Platform Configuration . . . . .	1102

# Chapter 18

## Module Documentation

### 18.1 Basic Crypto API methods (atcab\_)

These methods provide the most convenient, simple API to CryptoAuth chips.

#### Macros

- #define [atcab\\_cfg\\_discover\(...\)](#) [calib\\_cfg\\_discover\(\\_\\_VA\\_ARGS\\_\\_\)](#)
- #define [atcab\\_get\\_addr\(...\)](#) [calib\\_get\\_addr\(\\_\\_VA\\_ARGS\\_\\_\)](#)
- #define [atca\\_execute\\_command\(...\)](#) [calib\\_execute\\_command\(\\_\\_VA\\_ARGS\\_\\_\)](#)
- #define [SHA\\_CONTEXT\\_MAX\\_SIZE](#) (109)

#### Functions

- [ATCA\\_STATUS atcab\\_version](#) (char \*ver\_str)  
*basic API methods are all prefixed with atcab\_ (CryptoAuthLib Basic) the fundamental premise of the basic API is it is based on a single interface instance and that instance is global, so all basic API commands assume that one global device is the one to operate on.*
- [ATCA\\_STATUS atcab\\_init\\_ext](#) (ATCADevice \*device, ATCAIfaceCfg \*cfg)  
*Creates and initializes a ATCADevice context.*
- [ATCA\\_STATUS atcab\\_init](#) (ATCAIfaceCfg \*cfg)  
*Creates a global ATCADevice object used by Basic API.*
- [ATCA\\_STATUS atcab\\_init\\_device](#) (ATCADevice ca\_device)  
*Initialize the global ATCADevice object to point to one of your choosing for use with all the atcab\_ basic API.*
- [ATCA\\_STATUS atcab\\_release\\_ext](#) (ATCADevice \*device)  
*release (free) the an ATCADevice instance.*
- [ATCA\\_STATUS atcab\\_release](#) (void)  
*release (free) the global ATCADevice instance. This must be called in order to release or free up the interface.*
- [ATCADevice atcab\\_get\\_device](#) (void)  
*Get the global device object.*
- [ATCADeviceType atcab\\_get\\_device\\_type\\_ext](#) (ATCADevice device)  
*Get the selected device type of rthe device context.*
- [ATCADeviceType atcab\\_get\\_device\\_type](#) (void)  
*Get the current device type configured for the global ATCADevice.*
- bool [atcab\\_is\\_ca\\_device](#) (ATCADeviceType dev\_type)

Check whether the device is cryptoauth device.

- bool [atcab\\_is\\_ta\\_device](#) ([ATCADeviceType](#) dev\_type)

Check whether the device is Trust Anchor device.

- [ATCA\\_STATUS atcab\\_aes\\_cbc\\_init\\_ext](#) ([ATCADevice](#) device, [atca\\_aes\\_cbc\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block, const [uint8\\_t](#) \*iv)

Initialize context for AES CBC operation.

- [ATCA\\_STATUS atcab\\_aes\\_cbc\\_init](#) ([atca\\_aes\\_cbc\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block, const [uint8\\_t](#) \*iv)

Initialize context for AES CBC operation.

- [ATCA\\_STATUS atcab\\_aes\\_cbc\\_encrypt\\_block\\_ext](#) ([atca\\_aes\\_cbc\\_ctx\\_t](#) \*ctx, const [uint8\\_t](#) \*plaintext, [uint8\\_t](#) \*ciphertext)
- [ATCA\\_STATUS atcab\\_aes\\_cbc\\_encrypt\\_block](#) ([atca\\_aes\\_cbc\\_ctx\\_t](#) \*ctx, const [uint8\\_t](#) \*plaintext, [uint8\\_t](#) \*ciphertext)

Encrypt a block of data using CBC mode and a key within the device. [atcab\\_aes\\_cbc\\_init\(\)](#) should be called before the first use of this function.

- [ATCA\\_STATUS atcab\\_aes\\_cbc\\_decrypt\\_block](#) ([atca\\_aes\\_cbc\\_ctx\\_t](#) \*ctx, const [uint8\\_t](#) \*ciphertext, [uint8\\_t](#) \*plaintext)

Decrypt a block of data using CBC mode and a key within the device. [atcab\\_aes\\_cbc\\_init\(\)](#) should be called before the first use of this function.

- [ATCA\\_STATUS atcab\\_aes\\_cmac\\_init\\_ext](#) ([ATCADevice](#) device, [atca\\_aes\\_cmac\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block)

Initialize a CMAC calculation using an AES-128 key in the device.

- [ATCA\\_STATUS atcab\\_aes\\_cmac\\_init](#) ([atca\\_aes\\_cmac\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block)

Initialize a CMAC calculation using an AES-128 key in the device.

- [ATCA\\_STATUS atcab\\_aes\\_cmac\\_update](#) ([atca\\_aes\\_cmac\\_ctx\\_t](#) \*ctx, const [uint8\\_t](#) \*data, [uint32\\_t](#) data\_size)

Add data to an initialized CMAC calculation.

- [ATCA\\_STATUS atcab\\_aes\\_cmac\\_finish](#) ([atca\\_aes\\_cmac\\_ctx\\_t](#) \*ctx, [uint8\\_t](#) \*cmac, [uint32\\_t](#) cmac\_size)

Finish a CMAC operation returning the CMAC value.

- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_init\\_ext](#) ([ATCADevice](#) device, [atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block, [uint8\\_t](#) counter\_size, const [uint8\\_t](#) \*iv)

Initialize context for AES CTR operation with an existing IV, which is common when start a decrypt operation.

- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_init](#) ([atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block, [uint8\\_t](#) counter\_size, const [uint8\\_t](#) \*iv)

Initialize context for AES CTR operation with an existing IV, which is common when start a decrypt operation.

- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_init\\_rand\\_ext](#) ([ATCADevice](#) device, [atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block, [uint8\\_t](#) counter\_size, [uint8\\_t](#) \*iv)

Initialize context for AES CTR operation with a random nonce and counter set to 0 as the IV, which is common when starting an encrypt operation.

- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_init\\_rand](#) ([atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block, [uint8\\_t](#) counter\_size, [uint8\\_t](#) \*iv)

Initialize context for AES CTR operation with a random nonce and counter set to 0 as the IV, which is common when starting an encrypt operation.

- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_block](#) ([atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, const [uint8\\_t](#) \*input, [uint8\\_t](#) \*output)

Process a block of data using CTR mode and a key within the device. [atcab\\_aes\\_ctr\\_init\(\)](#) or [atcab\\_aes\\_ctr\\_init\\_rand\(\)](#) should be called before the first use of this function.

- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_encrypt\\_block](#) ([atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, const [uint8\\_t](#) \*plaintext, [uint8\\_t](#) \*ciphertext)

Encrypt a block of data using CTR mode and a key within the device. [atcab\\_aes\\_ctr\\_init\(\)](#) or [atcab\\_aes\\_ctr\\_init\\_rand\(\)](#) should be called before the first use of this function.

- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_decrypt\\_block](#) ([atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, const [uint8\\_t](#) \*ciphertext, [uint8\\_t](#) \*plaintext)

Decrypt a block of data using CTR mode and a key within the device. [atcab\\_aes\\_ctr\\_init\(\)](#) or [atcab\\_aes\\_ctr\\_init\\_rand\(\)](#) should be called before the first use of this function.

- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_increment](#) ([atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx)  
*Increments AES CTR counter value.*
- [ATCA\\_STATUS \\_atcab\\_exit](#) (void)
- [ATCA\\_STATUS atcab\\_wakeup](#) (void)  
*wakeup the CryptoAuth device*
- [ATCA\\_STATUS atcab\\_idle](#) (void)  
*idle the CryptoAuth device*
- [ATCA\\_STATUS atcab\\_sleep](#) (void)  
*invoke sleep on the CryptoAuth device*
- [ATCA\\_STATUS atcab\\_get\\_zone\\_size](#) (uint8\_t zone, uint16\_t slot, size\_t \*size)  
*Gets the size of the specified zone in bytes.*
- [ATCA\\_STATUS atcab\\_aes](#) (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*aes\_in, uint8\_t \*aes\_out)  
*Compute the AES-128 encrypt, decrypt, or GFM calculation.*
- [ATCA\\_STATUS atcab\\_aes\\_encrypt](#) (uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*plaintext, uint8\_t \*ciphertext)  
*Perform an AES-128 encrypt operation with a key in the device.*
- [ATCA\\_STATUS atcab\\_aes\\_encrypt\\_ext](#) ([ATCADevice](#) device, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*plaintext, uint8\_t \*ciphertext)  
*Perform an AES-128 encrypt operation with a key in the device.*
- [ATCA\\_STATUS atcab\\_aes\\_decrypt](#) (uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*ciphertext, uint8\_t \*plaintext)  
*Perform an AES-128 decrypt operation with a key in the device.*
- [ATCA\\_STATUS atcab\\_aes\\_decrypt\\_ext](#) ([ATCADevice](#) device, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*ciphertext, uint8\_t \*plaintext)  
*Perform an AES-128 decrypt operation with a key in the device.*
- [ATCA\\_STATUS atcab\\_aes\\_gfm](#) (const uint8\_t \*h, const uint8\_t \*input, uint8\_t \*output)  
*Perform a Galois Field Multiply (GFM) operation.*
- [ATCA\\_STATUS atcab\\_aes\\_gcm\\_init](#) ([atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*iv, size\_t iv\_size)  
*Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.*
- [ATCA\\_STATUS atcab\\_aes\\_gcm\\_init\\_rand](#) ([atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, uint16\_t key\_id, uint8\_t key\_block, size\_t rand\_size, const uint8\_t \*free\_field, size\_t free\_field\_size, uint8\_t \*iv)  
*Initialize context for AES GCM operation with a IV composed of a random and optional fixed(free) field, which is common when starting an encrypt operation.*
- [ATCA\\_STATUS atcab\\_aes\\_gcm\\_aad\\_update](#) ([atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*aad, uint32\_t aad\_size)  
*Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.*
- [ATCA\\_STATUS atcab\\_aes\\_gcm\\_encrypt\\_update](#) ([atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*plaintext, uint32\_t plaintext\_size, uint8\_t \*ciphertext)  
*Encrypt data using GCM mode and a key within the ATECC608 device. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function.*
- [ATCA\\_STATUS atcab\\_aes\\_gcm\\_encrypt\\_finish](#) ([atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, uint8\_t \*tag, size\_t tag\_size)  
*Complete a GCM encrypt operation returning the authentication tag.*
- [ATCA\\_STATUS atcab\\_aes\\_gcm\\_decrypt\\_update](#) ([atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*ciphertext, uint32\_t ciphertext\_size, uint8\_t \*plaintext)  
*Decrypt data using GCM mode and a key within the ATECC608 device. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function.*
- [ATCA\\_STATUS atcab\\_aes\\_gcm\\_decrypt\\_finish](#) ([atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*tag, size\_t tag\_size, bool \*is\_verified)  
*Complete a GCM decrypt operation verifying the authentication tag.*
- [ATCA\\_STATUS atcab\\_checkmac](#) (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*challenge, const uint8\_t \*response, const uint8\_t \*other\_data)  
*Compares a MAC response with input values.*



- **ATCA\_STATUS atcab\_counter** (uint8\_t mode, uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Compute the Counter functions.*
- **ATCA\_STATUS atcab\_counter\_increment** (uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Increments one of the device's monotonic counters.*
- **ATCA\_STATUS atcab\_counter\_read** (uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Read one of the device's monotonic counters.*
- **ATCA\_STATUS atcab\_derivekey** (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*mac)  
*Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.*
- **ATCA\_STATUS atcab\_ecdh\_base** (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, uint8\_t \*out\_nonce)  
*Base function for generating premaster secret key using ECDH.*
- **ATCA\_STATUS atcab\_ecdh** (uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms)  
*ECDH command with a private key in a slot and the premaster secret is returned in the clear.*
- **ATCA\_STATUS atcab\_ecdh\_enc** (uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*read\_key, uint16\_t read\_key\_id, const uint8\_t num\_in[(20)])  
*ECDH command with a private key in a slot and the premaster secret is read from the next slot.*
- **ATCA\_STATUS atcab\_ecdh\_ioenc** (uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*io\_key)  
*ECDH command with a private key in a slot and the premaster secret is returned encrypted using the IO protection key.*
- **ATCA\_STATUS atcab\_ecdh\_tempkey** (const uint8\_t \*public\_key, uint8\_t \*pms)  
*ECDH command with a private key in TempKey and the premaster secret is returned in the clear.*
- **ATCA\_STATUS atcab\_ecdh\_tempkey\_ioenc** (const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*io\_key)  
*ECDH command with a private key in TempKey and the premaster secret is returned encrypted using the IO protection key.*
- **ATCA\_STATUS atcab\_gendig** (uint8\_t zone, uint16\_t key\_id, const uint8\_t \*other\_data, uint8\_t other\_data\_size)  
*Issues a GenDig command, which performs a SHA256 hash on the source data indicated by zone with the contents of TempKey. See the CryptoAuth datasheet for your chip to see what the values of zone correspond to.*
- **ATCA\_STATUS atcab\_genkey\_base** (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*other\_data, uint8\_t \*public\_key)  
*Issues GenKey command, which can generate a private key, compute a public key, nd/or compute a digest of a public key.*
- **ATCA\_STATUS atcab\_genkey** (uint16\_t key\_id, uint8\_t \*public\_key)  
*Issues GenKey command, which generates a new random private key in slot/handle and returns the public key.*
- **ATCA\_STATUS atcab\_get\_pubkey** (uint16\_t key\_id, uint8\_t \*public\_key)  
*Uses GenKey command to calculate the public key from an existing private key in a slot.*
- **ATCA\_STATUS atcab\_hmac** (uint8\_t mode, uint16\_t key\_id, uint8\_t \*digest)  
*Issues a HMAC command, which computes an HMAC/SHA-256 digest of a key stored in the device, a challenge, and other information on the device.*
- **ATCA\_STATUS atcab\_info\_base** (uint8\_t mode, uint16\_t param2, uint8\_t \*out\_data)  
*Issues an Info command, which return internal device information and can control GPIO and the persistent latch.*
- **ATCA\_STATUS atcab\_info** (uint8\_t \*revision)  
*Use the Info command to get the device revision (DevRev).*
- **ATCA\_STATUS atcab\_info\_set\_latch** (bool state)  
*Use the Info command to set the persistent latch state for an ATECC608 device.*
- **ATCA\_STATUS atcab\_info\_get\_latch** (bool \*state)  
*Use the Info command to get the persistent latch current state for an ATECC608 device.*
- **ATCA\_STATUS atcab\_kdf** (uint8\_t mode, uint16\_t key\_id, const uint32\_t details, const uint8\_t \*message, uint8\_t \*out\_data, uint8\_t \*out\_nonce)  
*Executes the KDF command, which derives a new key in PRF, AES, or HKDF modes.*
- **ATCA\_STATUS atcab\_lock** (uint8\_t mode, uint16\_t summary\_crc)

The Lock command prevents future modifications of the Configuration and/or Data and OTP zones. If the device is so configured, then this command can be used to lock individual data slots. This command fails if the designated area is already locked.

- [ATCA\\_STATUS atcab\\_lock\\_config\\_zone](#) (void)  
*Unconditionally (no CRC required) lock the config zone.*
- [ATCA\\_STATUS atcab\\_lock\\_config\\_zone\\_crc](#) (uint16\_t summary\_crc)  
*Lock the config zone with summary CRC.*
- [ATCA\\_STATUS atcab\\_lock\\_data\\_zone](#) (void)  
*Unconditionally (no CRC required) lock the data zone (slots and OTP). for CryptoAuth devices and lock the setup for Trust Anchor device.*
- [ATCA\\_STATUS atcab\\_lock\\_data\\_zone\\_crc](#) (uint16\_t summary\_crc)  
*Lock the data zone (slots and OTP) with summary CRC.*
- [ATCA\\_STATUS atcab\\_lock\\_data\\_slot](#) (uint16\_t slot)  
*Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (KeyConfig.Lockable=1) (for cryptoauth devices) or Lock an individual handle in shared data element on an Trust Anchor device (for Trust Anchor devices).*
- [ATCA\\_STATUS atcab\\_mac](#) (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*challenge, uint8\_t \*digest)  
*Executes MAC command, which computes a SHA-256 digest of a key stored in the device, a challenge, and other information on the device.*
- [ATCA\\_STATUS atcab\\_nonce\\_base](#) (uint8\_t mode, uint16\_t zero, const uint8\_t \*num\_in, uint8\_t \*rand\_out)  
*Executes Nonce command, which loads a random or fixed nonce/data into the device for use by subsequent commands.*
- [ATCA\\_STATUS atcab\\_nonce](#) (const uint8\_t \*num\_in)  
*Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.*
- [ATCA\\_STATUS atcab\\_nonce\\_load](#) (uint8\_t target, const uint8\_t \*num\_in, uint16\_t num\_in\_size)  
*Execute a Nonce command in pass-through mode to load one of the device's internal buffers with a fixed value.*
- [ATCA\\_STATUS atcab\\_nonce\\_rand](#) (const uint8\_t \*num\_in, uint8\_t \*rand\_out)  
*Execute a Nonce command to generate a random nonce combining a host nonce (num\_in) and a device random number.*
- [ATCA\\_STATUS atcab\\_challenge](#) (const uint8\_t \*num\_in)  
*Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.*
- [ATCA\\_STATUS atcab\\_challenge\\_seed\\_update](#) (const uint8\_t \*num\_in, uint8\_t \*rand\_out)  
*Execute a Nonce command to generate a random challenge combining a host nonce (num\_in) and a device random number.*
- [ATCA\\_STATUS atcab\\_priv\\_write](#) (uint16\_t key\_id, const uint8\_t priv\_key[36], uint16\_t write\_key\_id, const uint8\_t write\_key[32], const uint8\_t num\_in[(20)])  
*Executes PrivWrite command, to write externally generated ECC private keys into the device.*
- [ATCA\\_STATUS atcab\\_random](#) (uint8\_t \*rand\_out)  
*Executes Random command, which generates a 32 byte random number from the device.*
- [ATCA\\_STATUS atcab\\_random\\_ext](#) (ATCADevice device, uint8\_t \*rand\_out)  
*Executes Random command, which generates a 32 byte random number from the device.*
- [ATCA\\_STATUS atcab\\_read\\_zone](#) (uint8\_t zone, uint16\_t slot, uint8\_t block, uint8\_t offset, uint8\_t \*data, uint8\_t len)  
*Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.*
- [ATCA\\_STATUS atcab\\_is\\_locked](#) (uint8\_t zone, bool \*is\_locked)  
*Executes Read command, which reads the configuration zone to see if the specified zone is locked.*
- [ATCA\\_STATUS atcab\\_is\\_config\\_locked](#) (bool \*is\_locked)  
*This function check whether configuration zone is locked or not.*
- [ATCA\\_STATUS atcab\\_is\\_data\\_locked](#) (bool \*is\_locked)  
*This function check whether data/setup zone is locked or not.*
- [ATCA\\_STATUS atcab\\_is\\_slot\\_locked](#) (uint16\_t slot, bool \*is\_locked)  
*This function check whether slot/handle is locked or not.*

- **ATCA\_STATUS atcab\_read\_bytes\_zone** (uint8\_t zone, uint16\_t slot, size\_t offset, uint8\_t \*data, size\_t length)
 

*Used to read an arbitrary number of bytes from any zone configured for clear reads.*
- **ATCA\_STATUS atcab\_read\_serial\_number** (uint8\_t \*serial\_number)
 

*This function returns serial number of the device.*
- **ATCA\_STATUS atcab\_read\_pubkey** (uint16\_t slot, uint8\_t \*public\_key)
 

*Executes Read command to read an ECC P256 public key from a slot configured for clear reads.*
- **ATCA\_STATUS atcab\_read\_sig** (uint16\_t slot, uint8\_t \*sig)
 

*Executes Read command to read a 64 byte ECDSA P256 signature from a slot configured for clear reads.*
- **ATCA\_STATUS atcab\_read\_config\_zone** (uint8\_t \*config\_data)
 

*Executes Read command to read the complete device configuration zone.*
- **ATCA\_STATUS atcab\_cmp\_config\_zone** (uint8\_t \*config\_data, bool \*same\_config)
 

*Compares a specified configuration zone with the configuration zone currently on the device.*
- **ATCA\_STATUS atcab\_read\_enc** (uint16\_t key\_id, uint8\_t block, uint8\_t \*data, const uint8\_t \*enc\_key, const uint16\_t enc\_key\_id, const uint8\_t num\_in[(20)])
 

*Executes Read command on a slot configured for encrypted reads and decrypts the data to return it as plaintext.*
- **ATCA\_STATUS atcab\_secureboot** (uint8\_t mode, uint16\_t param2, const uint8\_t \*digest, const uint8\_t \*signature, uint8\_t \*mac)
 

*Executes Secure Boot command, which provides support for secure boot of an external MCU or MPU.*
- **ATCA\_STATUS atcab\_secureboot\_mac** (uint8\_t mode, const uint8\_t \*digest, const uint8\_t \*signature, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)
 

*Executes Secure Boot command with encrypted digest and validated MAC response using the IO protection key.*
- **ATCA\_STATUS atcab\_selftest** (uint8\_t mode, uint16\_t param2, uint8\_t \*result)
 

*Executes the SelfTest command, which performs a test of one or more of the cryptographic engines within the ATCC608 chip.*
- **ATCA\_STATUS atcab\_sha\_base** (uint8\_t mode, uint16\_t length, const uint8\_t \*data\_in, uint8\_t \*data\_out, uint16\_t \*data\_out\_size)
 

*Executes SHA command, which computes a SHA-256 or HMAC/SHA-256 digest for general purpose use by the host system.*
- **ATCA\_STATUS atcab\_sha\_start** (void)
 

*Executes SHA command to initialize SHA-256 calculation engine.*
- **ATCA\_STATUS atcab\_sha\_update** (const uint8\_t \*message)
 

*Executes SHA command to add 64 bytes of message data to the current context.*
- **ATCA\_STATUS atcab\_sha\_end** (uint8\_t \*digest, uint16\_t length, const uint8\_t \*message)
 

*Executes SHA command to complete SHA-256 or HMAC/SHA-256 operation.*
- **ATCA\_STATUS atcab\_sha\_read\_context** (uint8\_t \*context, uint16\_t \*context\_size)
 

*Executes SHA command to read the SHA-256 context back. Only for ATECC608 with SHA-256 contexts. HMAC not supported.*
- **ATCA\_STATUS atcab\_sha\_write\_context** (const uint8\_t \*context, uint16\_t context\_size)
 

*Executes SHA command to write (restore) a SHA-256 context into the the device. Only supported for ATECC608 with SHA-256 contexts.*
- **ATCA\_STATUS atcab\_sha** (uint16\_t length, const uint8\_t \*message, uint8\_t \*digest)
 

*Use the SHA command to compute a SHA-256 digest.*
- **ATCA\_STATUS atcab\_hw\_sha2\_256** (const uint8\_t \*data, size\_t data\_size, uint8\_t \*digest)
 

*Use the SHA command to compute a SHA-256 digest.*
- **ATCA\_STATUS atcab\_hw\_sha2\_256\_init** (atca\_sha256\_ctx\_t \*ctx)
 

*Initialize a SHA context for performing a hardware SHA-256 operation on a device. Note that only one SHA operation can be run at a time.*
- **ATCA\_STATUS atcab\_hw\_sha2\_256\_update** (atca\_sha256\_ctx\_t \*ctx, const uint8\_t \*data, size\_t data\_size)
 

*Add message data to a SHA context for performing a hardware SHA-256 operation on a device.*
- **ATCA\_STATUS atcab\_hw\_sha2\_256\_finish** (atca\_sha256\_ctx\_t \*ctx, uint8\_t \*digest)

- Finish SHA-256 digest for a SHA context for performing a hardware SHA-256 operation on a device.*

  - **ATCA\_STATUS atcab\_sha\_hmac\_init** (atca\_hmac\_sha256\_ctx\_t \*ctx, uint16\_t key\_slot)

*Executes SHA command to start an HMAC/SHA-256 operation.*
- **ATCA\_STATUS atcab\_sha\_hmac\_update** (atca\_hmac\_sha256\_ctx\_t \*ctx, const uint8\_t \*data, size\_t data\_size)

*Executes SHA command to add an arbitrary amount of message data to a HMAC/SHA-256 operation.*
- **ATCA\_STATUS atcab\_sha\_hmac\_finish** (atca\_hmac\_sha256\_ctx\_t \*ctx, uint8\_t \*digest, uint8\_t target)

*Executes SHA command to complete a HMAC/SHA-256 operation.*
- **ATCA\_STATUS atcab\_sha\_hmac** (const uint8\_t \*data, size\_t data\_size, uint16\_t key\_slot, uint8\_t \*digest, uint8\_t target)

*Use the SHA command to compute an HMAC/SHA-256 operation.*
- **ATCA\_STATUS atcab\_sign\_base** (uint8\_t mode, uint16\_t key\_id, uint8\_t \*signature)

*Executes the Sign command, which generates a signature using the ECDSA algorithm.*
- **ATCA\_STATUS atcab\_sign** (uint16\_t key\_id, const uint8\_t \*msg, uint8\_t \*signature)

*Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*
- **ATCA\_STATUS atcab\_sign\_internal** (uint16\_t key\_id, bool is\_invalidate, bool is\_full\_sn, uint8\_t \*signature)

*Executes Sign command to sign an internally generated message.*
- **ATCA\_STATUS atcab\_updateextra** (uint8\_t mode, uint16\_t new\_value)

*Executes UpdateExtra command to update the values of the two extra bytes within the Configuration zone (bytes 84 and 85).*
- **ATCA\_STATUS atcab\_verify** (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*public\_key, const uint8\_t \*other\_data, uint8\_t \*mac)

*Executes the Verify command, which takes an ECDSA [R,S] signature and verifies that it is correctly generated from a given message and public key. In all cases, the signature is an input to the command.*
- **ATCA\_STATUS atcab\_verify\_extern** (const uint8\_t \*message, const uint8\_t \*signature, const uint8\_t \*public\_key, bool \*is\_verified)

*Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*
- **ATCA\_STATUS atcab\_verify\_extern\_mac** (const uint8\_t \*message, const uint8\_t \*signature, const uint8\_t \*public\_key, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)

*Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. This function is only available on the ATECC608.*
- **ATCA\_STATUS atcab\_verify\_stored** (const uint8\_t \*message, const uint8\_t \*signature, uint16\_t key\_id, bool \*is\_verified)

*Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*
- **ATCA\_STATUS atcab\_verify\_stored\_mac** (const uint8\_t \*message, const uint8\_t \*signature, uint16\_t key\_id, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)

*Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with a public key stored in the device. This function is only available on the ATECC608.*
- **ATCA\_STATUS atcab\_verify\_validate** (uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*other\_data, bool \*is\_verified)

*Executes the Verify command in Validate mode to validate a public key stored in a slot.*
- **ATCA\_STATUS atcab\_verify\_invalidate** (uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*other\_data, bool \*is\_verified)

*Executes the Verify command in Invalidate mode which invalidates a previously validated public key stored in a slot.*
- **ATCA\_STATUS atcab\_write** (uint8\_t zone, uint16\_t address, const uint8\_t \*value, const uint8\_t \*mac)

*Executes the Write command, which writes either one four byte word or a 32-byte block to one of the EEPROM zones on the device. Depending upon the value of the WriteConfig byte for this slot, the data may be required to be encrypted by the system prior to being sent to the device. This command cannot be used to write slots configured as ECC private keys.*

- [ATCA\\_STATUS atcab\\_write\\_zone](#) (uint8\_t zone, uint16\_t slot, uint8\_t block, uint8\_t offset, const uint8\_t \*data, uint8\_t len)  
*Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.*
- [ATCA\\_STATUS atcab\\_write\\_bytes\\_zone](#) (uint8\_t zone, uint16\_t slot, size\_t offset\_bytes, const uint8\_t \*data, size\_t length)  
*Executes the Write command, which writes data into the configuration, otp, or data zones with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).*
- [ATCA\\_STATUS atcab\\_write\\_pubkey](#) (uint16\_t slot, const uint8\_t \*public\_key)  
*Uses the write command to write a public key to a slot in the proper format.*
- [ATCA\\_STATUS atcab\\_write\\_config\\_zone](#) (const uint8\_t \*config\_data)  
*Executes the Write command, which writes the configuration zone.*
- [ATCA\\_STATUS atcab\\_write\\_enc](#) (uint16\_t key\_id, uint8\_t block, const uint8\_t \*data, const uint8\_t \*enc\_key, const uint16\_t enc\_key\_id, const uint8\_t num\_in[(20)])  
*Executes the Write command, which performs an encrypted write of a 32 byte block into given slot.*
- [ATCA\\_STATUS atcab\\_write\\_config\\_counter](#) (uint16\_t counter\_id, uint32\_t counter\_value)  
*Initialize one of the monotonic counters in device with a specific value.*

## Variables

- `SHARED_LIB_IMPORT` [ATCADevice\\_gDevice](#)
- [ATCA\\_STATUS atcab\\_printbin](#) (uint8\_t \*binary, size\_t bin\_len, bool add\_space)
- `const char *` [atca\\_basic\\_aes\\_gcm\\_version](#)
- [ATCA\\_STATUS calib\\_aes\\_gcm\\_init](#) ([ATCADevice](#) device, [atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, uint16\_t key\_id, uint8\_t \*key\_block, const uint8\_t \*iv, size\_t iv\_size)  
*Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.*
- [ATCA\\_STATUS calib\\_aes\\_gcm\\_init\\_rand](#) ([ATCADevice](#) device, [atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, uint16\_t key\_id, uint8\_t \*key\_block, size\_t rand\_size, const uint8\_t \*free\_field, size\_t free\_field\_size, uint8\_t \*iv)  
*Initialize context for AES GCM operation with a IV composed of a random and optional fixed(free) field, which is common when starting an encrypt operation.*
- [ATCA\\_STATUS calib\\_aes\\_gcm\\_aad\\_update](#) ([ATCADevice](#) device, [atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*aad, uint32\_t aad\_size)  
*Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.*
- [ATCA\\_STATUS calib\\_aes\\_gcm\\_encrypt\\_update](#) ([ATCADevice](#) device, [atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*plaintext, uint32\_t plaintext\_size, uint8\_t \*ciphertext)  
*Encrypt data using GCM mode and a key within the ATECC608 device. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function.*
- [ATCA\\_STATUS calib\\_aes\\_gcm\\_encrypt\\_finish](#) ([ATCADevice](#) device, [atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, uint8\_t \*tag, size\_t tag\_size)  
*Complete a GCM encrypt operation returning the authentication tag.*
- [ATCA\\_STATUS calib\\_aes\\_gcm\\_decrypt\\_update](#) ([ATCADevice](#) device, [atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*ciphertext, uint32\_t ciphertext\_size, uint8\_t \*plaintext)  
*Decrypt data using GCM mode and a key within the ATECC608 device. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function.*
- [ATCA\\_STATUS calib\\_aes\\_gcm\\_decrypt\\_finish](#) ([ATCADevice](#) device, [atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*tag, size\_t tag\_size, bool \*is\_verified)  
*Complete a GCM decrypt operation verifying the authentication tag.*
- `#define` [ATCA\\_AES\\_GCM\\_IV\\_STD\\_LENGTH](#) 12
- `typedef struct` [atca\\_aes\\_gcm\\_ctx](#) [atca\\_aes\\_gcm\\_ctx\\_t](#)

### 18.1.1 Detailed Description

These methods provide the most convenient, simple API to CryptoAuth chips.

### 18.1.2 Macro Definition Documentation

#### 18.1.2.1 ATCA\_AES\_GCM\_IV\_STD\_LENGTH

```
#define ATCA_AES_GCM_IV_STD_LENGTH 12
```

#### 18.1.2.2 atca\_execute\_command

```
#define atca_execute_command(  
    ... ) calib_execute_command(__VA_ARGS__)
```

#### 18.1.2.3 atcab\_cfg\_discover

```
#define atcab_cfg_discover(  
    ... ) calib_cfg_discover(__VA_ARGS__)
```

#### 18.1.2.4 atcab\_get\_addr

```
#define atcab_get_addr(  
    ... ) calib_get_addr(__VA_ARGS__)
```

#### 18.1.2.5 SHA\_CONTEXT\_MAX\_SIZE

```
#define SHA_CONTEXT_MAX_SIZE (109)
```

### 18.1.3 Typedef Documentation

### 18.1.3.1 atca\_aes\_gcm\_ctx\_t

```
typedef struct atca_aes_gcm_ctx atca_aes_gcm_ctx_t
```

Context structure for AES GCM operations.

## 18.1.4 Function Documentation

### 18.1.4.1 \_atcab\_exit()

```
ATCA_STATUS _atcab_exit (
    void )
```

### 18.1.4.2 atcab\_aes()

```
ATCA_STATUS atcab_aes (
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * aes_in,
    uint8_t * aes_out )
```

Compute the AES-128 encrypt, decrypt, or GFM calculation.

#### Parameters

in	<i>mode</i>	The mode for the AES command.
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>aes_in</i>	Input data to the AES command (16 bytes).
out	<i>aes_out</i>	Output data from the AES command is returned here (16 bytes).

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.3 atcab\_aes\_cbc\_decrypt\_block()

```
ATCA_STATUS atcab_aes_cbc_decrypt_block (
    atca_aes_cbc_ctx_t * ctx,
    const uint8_t * ciphertext,
    uint8_t * plaintext )
```

Decrypt a block of data using CBC mode and a key within the device. [atcab\\_aes\\_cbc\\_init\(\)](#) should be called before the first use of this function.



## 18.1 Basic Crypto API methods (atcab\_)

---

### Parameters

in	<i>ctx</i>	AES CBC context.
in	<i>ciphertext</i>	Ciphertext to be decrypted (16 bytes).
out	<i>plaintext</i>	Decrypted data is returned here (16 bytes).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.4 atcab\_aes\_cbc\_encrypt\_block()

```
ATCA_STATUS atcab_aes_cbc_encrypt_block (
    atca_aes_cbc_ctx_t * ctx,
    const uint8_t * plaintext,
    uint8_t * ciphertext )
```

Encrypt a block of data using CBC mode and a key within the device. [atcab\\_aes\\_cbc\\_init\(\)](#) should be called before the first use of this function.

### Parameters

in	<i>ctx</i>	AES CBC context.
in	<i>plaintext</i>	Plaintext to be encrypted (16 bytes).
out	<i>ciphertext</i>	Encrypted data is returned here (16 bytes).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.5 atcab\_aes\_cbc\_encrypt\_block\_ext()

```
ATCA_STATUS atcab_aes_cbc_encrypt_block_ext (
    atca_aes_cbc_ctx_t * ctx,
    const uint8_t * plaintext,
    uint8_t * ciphertext )
```

#### 18.1.4.6 atcab\_aes\_cbc\_init()

```
ATCA_STATUS atcab_aes_cbc_init (
    atca_aes_cbc_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * iv )
```

Initialize context for AES CBC operation.



**Parameters**

in	<i>ctx</i>	AES CBC context to be initialized
in	<i>key_id</i>	Key location. Can either be a slot/handles or in TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>iv</i>	Initialization vector (16 bytes).

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.7 atcab\_aes\_cbc\_init\_ext()**

```
ATCA_STATUS atcab_aes_cbc_init_ext (
    ATCADevice device,
    atca_aes_cbc_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * iv )
```

Initialize context for AES CBC operation.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES CBC context to be initialized
in	<i>key_id</i>	Key location. Can either be a slot/handles or in TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>iv</i>	Initialization vector (16 bytes).

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.8 atcab\_aes\_cmac\_finish()**

```
ATCA_STATUS atcab_aes_cmac_finish (
    atca_aes_cmac_ctx_t * ctx,
    uint8_t * cmac,
    uint32_t cmac_size )
```

Finish a CMAC operation returning the CMAC value.

### Parameters

in	<i>ctx</i>	AES-128 CMAC context.
out	<i>cmac</i>	CMAC is returned here.
in	<i>cmac_size</i>	Size of CMAC requested in bytes (max 16 bytes).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.9 atcab\_aes\_cmac\_init()

```
ATCA_STATUS atcab_aes_cmac_init (
    atca_aes_cmac_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block )
```

Initialize a CMAC calculation using an AES-128 key in the device.

### Parameters

in	<i>ctx</i>	AES-128 CMAC context.
in	<i>key_id</i>	Key location. Can either be a slot/handles or in TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.10 atcab\_aes\_cmac\_init\_ext()

```
ATCA_STATUS atcab_aes_cmac_init_ext (
    ATCADevice device,
    atca_aes_cmac_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block )
```

Initialize a CMAC calculation using an AES-128 key in the device.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES-128 CMAC context.
in	<i>key_id</i>	Key location. Can either be a slot/handles or in TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.11 atcab\_aes\_cmac\_update()**

```
ATCA_STATUS atcab_aes_cmac_update (
    atca_aes_cmac_ctx_t * ctx,
    const uint8_t * data,
    uint32_t data_size )
```

Add data to an initialized CMAC calculation.

**Parameters**

in	<i>ctx</i>	AES-128 CMAC context.
in	<i>data</i>	Data to be added.
in	<i>data_size</i>	Size of the data to be added in bytes.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.12 atcab\_aes\_ctr\_block()**

```
ATCA_STATUS atcab_aes_ctr_block (
    atca_aes_ctr_ctx_t * ctx,
    const uint8_t * input,
    uint8_t * output )
```

Process a block of data using CTR mode and a key within the device. [atcab\\_aes\\_ctr\\_init\(\)](#) or [atcab\\_aes\\_ctr\\_init\\_rand\(\)](#) should be called before the first use of this function.

**Parameters**

in	<i>ctx</i>	AES CTR context structure.
in	<i>input</i>	Input data to be processed (16 bytes).
out	<i>output</i>	Output data is returned here (16 bytes).

**Returns**

ATCA\_SUCCESS on success, ATCA\_INVALID\_SIZE on counter overflow, otherwise an error code.

### 18.1.4.13 atcab\_aes\_ctr\_decrypt\_block()

```
ATCA_STATUS atcab_aes_ctr_decrypt_block (
    atca_aes_ctr_ctx_t * ctx,
    const uint8_t * ciphertext,
    uint8_t * plaintext )
```

Decrypt a block of data using CTR mode and a key within the device. [atcab\\_aes\\_ctr\\_init\(\)](#) or [atcab\\_aes\\_ctr\\_init\\_rand\(\)](#) should be called before the first use of this function.

#### Parameters

in	<i>ctx</i>	AES CTR context structure.
in	<i>ciphertext</i>	Ciphertext to be decrypted (16 bytes).
out	<i>plaintext</i>	Decrypted data is returned here (16 bytes).

#### Returns

ATCA\_SUCCESS on success, ATCA\_INVALID\_SIZE on counter overflow, otherwise an error code.

### 18.1.4.14 atcab\_aes\_ctr\_encrypt\_block()

```
ATCA_STATUS atcab_aes_ctr_encrypt_block (
    atca_aes_ctr_ctx_t * ctx,
    const uint8_t * plaintext,
    uint8_t * ciphertext )
```

Encrypt a block of data using CTR mode and a key within the device. [atcab\\_aes\\_ctr\\_init\(\)](#) or [atcab\\_aes\\_ctr\\_init\\_rand\(\)](#) should be called before the first use of this function.

#### Parameters

in	<i>ctx</i>	AES CTR context structure.
in	<i>plaintext</i>	Plaintext to be encrypted (16 bytes).
out	<i>ciphertext</i>	Encrypted data is returned here (16 bytes).

#### Returns

ATCA\_SUCCESS on success, ATCA\_INVALID\_SIZE on counter overflow, otherwise an error code.

### 18.1.4.15 atcab\_aes\_ctr\_increment()

```
ATCA_STATUS atcab_aes_ctr_increment (
    atca_aes_ctr_ctx_t * ctx )
```

Increments AES CTR counter value.

## Parameters

<i>in, out</i>	<i>ctx</i>	AES CTR context
----------------	------------	-----------------

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 18.1.4.16 atcab\_aes\_ctr\_init()

```
ATCA_STATUS atcab_aes_ctr_init (
    atca_aes_ctr_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block,
    uint8_t counter_size,
    const uint8_t * iv )
```

Initialize context for AES CTR operation with an existing IV, which is common when start a decrypt operation.

The IV is a combination of nonce (left-field) and big-endian counter (right-field). The counter\_size field sets the size of the counter and the remaining bytes are assumed to be the nonce.

## Parameters

in	<i>ctx</i>	AES CTR context to be initialized.
in	<i>key_id</i>	Key location. Can either be a slot/handles or in TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>counter_size</i>	Size of counter in IV in bytes. 4 bytes is a common size.
in	<i>iv</i>	Initialization vector (concatenation of nonce and counter) 16 bytes.

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 18.1.4.17 atcab\_aes\_ctr\_init\_ext()

```
ATCA_STATUS atcab_aes_ctr_init_ext (
    ATCADevice device,
    atca_aes_ctr_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block,
    uint8_t counter_size,
    const uint8_t * iv )
```

Initialize context for AES CTR operation with an existing IV, which is common when start a decrypt operation.

The IV is a combination of nonce (left-field) and big-endian counter (right-field). The counter\_size field sets the size of the counter and the remaining bytes are assumed to be the nonce.

## 18.1 Basic Crypto API methods (atcab\_)

---

### Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES CTR context to be initialized.
in	<i>key_id</i>	Key location. Can either be a slot/handles or in TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>counter_size</i>	Size of counter in IV in bytes. 4 bytes is a common size.
in	<i>iv</i>	Initialization vector (concatenation of nonce and counter) 16 bytes.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.18 atcab\_aes\_ctr\_init\_rand()

```
ATCA_STATUS atcab_aes_ctr_init_rand (
    atca_aes_ctr_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block,
    uint8_t counter_size,
    uint8_t * iv )
```

Initialize context for AES CTR operation with a random nonce and counter set to 0 as the IV, which is common when starting an encrypt operation.

The IV is a combination of nonce (left-field) and big-endian counter (right-field). The counter\_size field sets the size of the counter and the remaining bytes are assumed to be the nonce.

### Parameters

in	<i>ctx</i>	AES CTR context to be initialized.
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>counter_size</i>	Size of counter in IV in bytes. 4 bytes is a common size.
out	<i>iv</i>	Initialization vector (concatenation of nonce and counter) is returned here (16 bytes).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.19 atcab\_aes\_ctr\_init\_rand\_ext()

```
ATCA_STATUS atcab_aes_ctr_init_rand_ext (
    ATCADevice device,
```

```

    atca_aes_ctr_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block,
    uint8_t counter_size,
    uint8_t * iv )

```

Initialize context for AES CTR operation with a random nonce and counter set to 0 as the IV, which is common when starting an encrypt operation.

The IV is a combination of nonce (left-field) and big-endian counter (right-field). The counter\_size field sets the size of the counter and the remaining bytes are assumed to be the nonce.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES CTR context to be initialized.
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>counter_size</i>	Size of counter in IV in bytes. 4 bytes is a common size.
out	<i>iv</i>	Initialization vector (concatenation of nonce and counter) is returned here (16 bytes).

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.20 atcab\_aes\_decrypt()

```

ATCA_STATUS atcab_aes_decrypt (
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * ciphertext,
    uint8_t * plaintext )

```

Perform an AES-128 decrypt operation with a key in the device.

#### Parameters

in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>ciphertext</i>	Input ciphertext to be decrypted (16 bytes).
out	<i>plaintext</i>	Output plaintext is returned here (16 bytes).

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.21 atcab\_aes\_decrypt\_ext()

```
ATCA_STATUS atcab_aes_decrypt_ext (
    ATCADevice device,
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * ciphertext,
    uint8_t * plaintext )
```

Perform an AES-128 decrypt operation with a key in the device.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>ciphertext</i>	Input ciphertext to be decrypted (16 bytes).
out	<i>plaintext</i>	Output plaintext is returned here (16 bytes).

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.22 atcab\_aes\_encrypt()

```
ATCA_STATUS atcab_aes_encrypt (
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * plaintext,
    uint8_t * ciphertext )
```

Perform an AES-128 encrypt operation with a key in the device.

#### Parameters

in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>plaintext</i>	Input plaintext to be encrypted (16 bytes).
out	<i>ciphertext</i>	Output ciphertext is returned here (16 bytes).

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.



#### 18.1.4.23 atcab\_aes\_encrypt\_ext()

```
ATCA_STATUS atcab_aes_encrypt_ext (
    ATCADevice device,
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * plaintext,
    uint8_t * ciphertext )
```

Perform an AES-128 encrypt operation with a key in the device.

##### Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>plaintext</i>	Input plaintext to be encrypted (16 bytes).
out	<i>ciphertext</i>	Output ciphertext is returned here (16 bytes).

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.24 atcab\_aes\_gcm\_aad\_update()

```
ATCA_STATUS atcab_aes_gcm_aad_update (
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * aad,
    uint32_t aad_size )
```

Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.

This can be called multiple times. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function. When there is AAD to include, this should be called before [atcab\\_aes\\_gcm\\_encrypt\\_update\(\)](#) or [atcab\\_aes\\_gcm\\_decrypt\\_update\(\)](#).

##### Parameters

in	<i>ctx</i>	AES GCM context
in	<i>aad</i>	Additional authenticated data to be added
in	<i>aad_size</i>	Size of aad in bytes

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.25 atcab\_aes\_gcm\_decrypt\_finish()

```
ATCA_STATUS atcab_aes_gcm_decrypt_finish (
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * tag,
    size_t tag_size,
    bool * is_verified )
```

Complete a GCM decrypt operation verifying the authentication tag.

#### Parameters

in	<i>ctx</i>	AES GCM context structure.
in	<i>tag</i>	Expected authentication tag.
in	<i>tag_size</i>	Size of tag in bytes (12 to 16 bytes).
out	<i>is_verified</i>	Returns whether or not the tag verified.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.26 atcab\_aes\_gcm\_decrypt\_update()

```
ATCA_STATUS atcab_aes_gcm_decrypt_update (
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * ciphertext,
    uint32_t ciphertext_size,
    uint8_t * plaintext )
```

Decrypt data using GCM mode and a key within the ATECC608 device. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function.

#### Parameters

in	<i>ctx</i>	AES GCM context structure.
in	<i>ciphertext</i>	Ciphertext to be decrypted.
in	<i>ciphertext_size</i>	Size of ciphertext in bytes.
out	<i>plaintext</i>	Decrypted data is returned here.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.27 atcab\_aes\_gcm\_encrypt\_finish()

```
ATCA_STATUS atcab_aes_gcm_encrypt_finish (
    atca_aes_gcm_ctx_t * ctx,
```

```
uint8_t * tag,
size_t tag_size )
```

Complete a GCM encrypt operation returning the authentication tag.

#### Parameters

in	<i>ctx</i>	AES GCM context structure.
out	<i>tag</i>	Authentication tag is returned here.
in	<i>tag_size</i>	Tag size in bytes (12 to 16 bytes).

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.28 atcab\_aes\_gcm\_encrypt\_update()

```
ATCA_STATUS atcab_aes_gcm_encrypt_update (
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * plaintext,
    uint32_t plaintext_size,
    uint8_t * ciphertext )
```

Encrypt data using GCM mode and a key within the ATECC608 device. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function.

#### Parameters

in	<i>ctx</i>	AES GCM context structure.
in	<i>plaintext</i>	Plaintext to be encrypted (16 bytes).
in	<i>plaintext_size</i>	Size of plaintext in bytes.
out	<i>ciphertext</i>	Encrypted data is returned here.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.29 atcab\_aes\_gcm\_init()

```
ATCA_STATUS atcab_aes_gcm_init (
    atca_aes_gcm_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * iv,
    size_t iv_size )
```

Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.

## 18.1 Basic Crypto API methods (atcab\_)

### Parameters

in	<i>ctx</i>	AES GCM context to be initialized.
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>iv</i>	Initialization vector.
in	<i>iv_size</i>	Size of IV in bytes. Standard is 12 bytes.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.30 atcab\_aes\_gcm\_init\_rand()

```
ATCA_STATUS atcab_aes_gcm_init_rand (
    atca_aes_gcm_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block,
    size_t rand_size,
    const uint8_t * free_field,
    size_t free_field_size,
    uint8_t * iv )
```

Initialize context for AES GCM operation with a IV composed of a random and optional fixed(free) field, which is common when starting an encrypt operation.

### Parameters

in	<i>ctx</i>	AES CTR context to be initialized.
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>rand_size</i>	Size of the random field in bytes. Minimum and recommended size is 12 bytes. Max is 32 bytes.
in	<i>free_field</i>	Fixed data to include in the IV after the random field. Can be NULL if not used.
in	<i>free_field_size</i>	Size of the free field in bytes.
out	<i>iv</i>	Initialization vector is returned here. Its size will be rand_size and free_field_size combined.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.31 atcab\_aes\_gfm()

```
ATCA_STATUS atcab_aes_gfm (
    const uint8_t * h,
```

```
const uint8_t * input,
uint8_t * output )
```

Perform a Galois Field Multiply (GFM) operation.

#### Parameters

in	<i>h</i>	First input value (16 bytes).
in	<i>input</i>	Second input value (16 bytes).
out	<i>output</i>	GFM result is returned here (16 bytes).

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.32 atcab\_challenge()

```
ATCA_STATUS atcab_challenge (
    const uint8_t * num_in )
```

Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.

#### Parameters

in	<i>num_in</i>	Data to be loaded into TempKey (32 bytes).
----	---------------	--

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.33 atcab\_challenge\_seed\_update()

```
ATCA_STATUS atcab_challenge_seed_update (
    const uint8_t * num_in,
    uint8_t * rand_out )
```

Execute a Nonce command to generate a random challenge combining a host nonce (num\_in) and a device random number.

#### Parameters

in	<i>num_in</i>	Host nonce to be combined with the device random number (20 bytes).
out	<i>rand_out</i>	Internally generated 32-byte random number that was used in the nonce/challenge calculation is returned here. Can be NULL if not needed.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.34 atcab\_checkmac()

```
ATCA_STATUS atcab_checkmac (
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * challenge,
    const uint8_t * response,
    const uint8_t * other_data )
```

Compares a MAC response with input values.

### Parameters

in	<i>mode</i>	Controls which fields within the device are used in the message
in	<i>key_id</i>	Key location in the CryptoAuth device to use for the MAC
in	<i>challenge</i>	Challenge data (32 bytes)
in	<i>response</i>	MAC response data (32 bytes)
in	<i>other_data</i>	OtherData parameter (13 bytes)

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.35 atcab\_cmp\_config\_zone()

```
ATCA_STATUS atcab_cmp_config_zone (
    uint8_t * config_data,
    bool * same_config )
```

Compares a specified configuration zone with the configuration zone currently on the device.

This only compares the static portions of the configuration zone and skips those that are unique per device (first 16 bytes) and areas that can change after the configuration zone has been locked (e.g. LastKeyUse).

### Parameters

in	<i>config_data</i>	Full configuration data to compare the device against.
out	<i>same_config</i>	Result is returned here. True if the static portions on the configuration zones are the same.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.36 atcab\_counter()**

```
ATCA_STATUS atcab_counter (
    uint8_t mode,
    uint16_t counter_id,
    uint32_t * counter_value )
```

Compute the Counter functions.

**Parameters**

in	<i>mode</i>	the mode used for the counter
in	<i>counter_id</i>	The counter to be used
out	<i>counter_value</i>	pointer to the counter value returned from device

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.37 atcab\_counter\_increment()**

```
ATCA_STATUS atcab_counter_increment (
    uint16_t counter_id,
    uint32_t * counter_value )
```

Increments one of the device's monotonic counters.

**Parameters**

in	<i>counter_id</i>	Counter to be incremented
out	<i>counter_value</i>	New value of the counter is returned here. Can be NULL if not needed.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.38 atcab\_counter\_read()**

```
ATCA_STATUS atcab_counter_read (
    uint16_t counter_id,
    uint32_t * counter_value )
```

## 18.1 Basic Crypto API methods (atcab\_)

---

Read one of the device's monotonic counters.

### Parameters

in	<i>counter_id</i>	Counter to be read
out	<i>counter_value</i>	Counter value is returned here.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.39 atcab\_derivekey()

```
ATCA_STATUS atcab_derivekey (
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * mac )
```

Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.

### Parameters

in	<i>mode</i>	Bit 2 must match the value in TempKey.SourceFlag
in	<i>key_id</i>	Key slot to be written
in	<i>mac</i>	Optional 32 byte MAC used to validate operation. NULL if not required.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.40 atcab\_ecdh()

```
ATCA_STATUS atcab_ecdh (
    uint16_t key_id,
    const uint8_t * public_key,
    uint8_t * pms )
```

ECDH command with a private key in a slot and the premaster secret is returned in the clear.

### Parameters

in	<i>key_id</i>	Slot of private key for ECDH computation
in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	<i>pms</i>	Computed ECDH premaster secret is returned here. 32 bytes.



**Returns**

ATCA\_SUCCESS on success

**18.1.4.41 atcab\_ecdh\_base()**

```
ATCA_STATUS atcab_ecdh_base (
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * public_key,
    uint8_t * pms,
    uint8_t * out_nonce )
```

Base function for generating premaster secret key using ECDH.

**Parameters**

in	<i>mode</i>	Mode to be used for ECDH computation
in	<i>key_id</i>	Slot of key for ECDH computation
in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	<i>pms</i>	Computed ECDH pre-master secret is returned here (32 bytes) if returned directly. Otherwise NULL.
out	<i>out_nonce</i>	Nonce used to encrypt pre-master secret. NULL if output encryption not used.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.42 atcab\_ecdh\_enc()**

```
ATCA_STATUS atcab_ecdh_enc (
    uint16_t key_id,
    const uint8_t * public_key,
    uint8_t * pms,
    const uint8_t * read_key,
    uint16_t read_key_id,
    const uint8_t num_in[ (20) ] )
```

ECDH command with a private key in a slot and the premaster secret is read from the next slot.

This function only works for even numbered slots with the proper configuration.

**Parameters**

in	<i>key_id</i>	Slot of key for ECDH computation
in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.

## 18.1 Basic Crypto API methods (atcab\_)

### Parameters

out	<i>pms</i>	Computed ECDH premaster secret is returned here (32 bytes).
in	<i>read_key</i>	Read key for the premaster secret slot (key_id 1).
in	<i>read_key_id</i>	Read key slot for read_key.
in	<i>num_in</i>	20 byte host nonce to inject into Nonce calculation

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.43 atcab\_ecdh\_ioenc()

```
ATCA_STATUS atcab_ecdh_ioenc (
    uint16_t key_id,
    const uint8_t * public_key,
    uint8_t * pms,
    const uint8_t * io_key )
```

ECDH command with a private key in a slot and the premaster secret is returned encrypted using the IO protection key.

### Parameters

in	<i>key_id</i>	Slot of key for ECDH computation
in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	<i>pms</i>	Computed ECDH premaster secret is returned here (32 bytes).
in	<i>io_key</i>	IO protection key.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.44 atcab\_ecdh\_tempkey()

```
ATCA_STATUS atcab_ecdh_tempkey (
    const uint8_t * public_key,
    uint8_t * pms )
```

ECDH command with a private key in TempKey and the premaster secret is returned in the clear.

## Parameters

in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	<i>pms</i>	Computed ECDH premaster secret is returned here (32 bytes).

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 18.1.4.45 atcab\_ecdh\_tempkey\_ioenc()

```
ATCA_STATUS atcab_ecdh_tempkey_ioenc (
    const uint8_t * public_key,
    uint8_t * pms,
    const uint8_t * io_key )
```

ECDH command with a private key in TempKey and the premaster secret is returned encrypted using the IO protection key.

## Parameters

in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	<i>pms</i>	Computed ECDH premaster secret is returned here (32 bytes).
in	<i>io_key</i>	IO protection key.

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 18.1.4.46 atcab\_gendig()

```
ATCA_STATUS atcab_gendig (
    uint8_t zone,
    uint16_t key_id,
    const uint8_t * other_data,
    uint8_t other_data_size )
```

Issues a GenDig command, which performs a SHA256 hash on the source data indicated by zone with the contents of TempKey. See the CryptoAuth datasheet for your chip to see what the values of zone correspond to.

## Parameters

in	<i>zone</i>	Designates the source of the data to hash with TempKey.
in	<i>key_id</i>	Indicates the key, OTP block, or message order for shared nonce mode.
in	<i>other_data</i>	Four bytes of data for SHA calculation when using a NoMac key, 32 bytes for "Shared Nonce" mode, otherwise ignored (can be NULL).
in	<i>other_data_size</i>	Size of other_data in bytes.

## 18.1 Basic Crypto API methods (atcab\_)

---

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.47 atcab\_genkey()

```
ATCA_STATUS atcab_genkey (
    uint16_t key_id,
    uint8_t * public_key )
```

Issues GenKey command, which generates a new random private key in slot/handle and returns the public key.

### Parameters

in	<i>key_id</i>	Slot number where an ECC private key is configured. Can also be ATCA_TEMPKEY_KEYID to generate a private key in TempKey.
out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Set to NULL if public key isn't required.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.48 atcab\_genkey\_base()

```
ATCA_STATUS atcab_genkey_base (
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * other_data,
    uint8_t * public_key )
```

Issues GenKey command, which can generate a private key, compute a public key, nd/or compute a digest of a public key.

### Parameters

in	<i>mode</i>	Mode determines what operations the GenKey command performs.
in	<i>key_id</i>	Slot to perform the GenKey command on.
in	<i>other_data</i>	OtherData for PubKey digest calculation. Can be set to NULL otherwise.
out	<i>public_key</i>	If the mode indicates a public key will be calculated, it will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Set to NULL if public key isn't required.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.49 atcab\_get\_device()

```
ATCADevice atcab_get_device (
    void )
```

Get the global device object.

##### Returns

instance of global ATCADevice

#### 18.1.4.50 atcab\_get\_device\_type()

```
ATCADeviceType atcab_get_device_type (
    void )
```

Get the current device type configured for the global ATCADevice.

##### Returns

Device type if basic api is initialized or ATCA\_DEV\_UNKNOWN.

#### 18.1.4.51 atcab\_get\_device\_type\_ext()

```
ATCADeviceType atcab_get_device_type_ext (
    ATCADevice device )
```

Get the selected device type of the device context.

##### Parameters

in	<i>device</i>	Device context pointer
----	---------------	------------------------

##### Returns

Device type if basic api is initialized or ATCA\_DEV\_UNKNOWN.

#### 18.1.4.52 atcab\_get\_pubkey()

```
ATCA_STATUS atcab_get_pubkey (
    uint16_t key_id,
    uint8_t * public_key )
```

## 18.1 Basic Crypto API methods (atcab\_)

Uses GenKey command to calculate the public key from an existing private key in a slot.

### Parameters

in	<i>key_id</i>	Slot number of the private key.
out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Set to NULL if public key isn't required.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.53 atcab\_get\_zone\_size()

```
ATCA_STATUS atcab_get_zone_size (
    uint8_t zone,
    uint16_t slot,
    size_t * size )
```

Gets the size of the specified zone in bytes.

### Parameters

in	<i>zone</i>	Zone to get size information from. Config(0), OTP(1), or Data(2) which requires a slot.
in	<i>slot</i>	If zone is Data(2), the slot to query for size.
out	<i>size</i>	Zone size is returned here.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.54 atcab\_hmac()

```
ATCA_STATUS atcab_hmac (
    uint8_t mode,
    uint16_t key_id,
    uint8_t * digest )
```

Issues a HMAC command, which computes an HMAC/SHA-256 digest of a key stored in the device, a challenge, and other information on the device.

### Parameters

in	<i>mode</i>	Controls which fields within the device are used in the message.
in	<i>key_id</i>	Which key is to be used to generate the response. Bits 0:3 only are used to select a slot but all 16 bits are used in the HMAC message.
out	<i>digest</i>	HMAC digest is returned in this buffer (32 bytes).

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.55 atcab\_hw\_sha2\_256()**

```
ATCA_STATUS atcab_hw_sha2_256 (
    const uint8_t * data,
    size_t data_size,
    uint8_t * digest )
```

Use the SHA command to compute a SHA-256 digest.

**Parameters**

in	<i>data</i>	Message data to be hashed.
in	<i>data_size</i>	Size of data in bytes.
out	<i>digest</i>	Digest is returned here (32 bytes).

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.56 atcab\_hw\_sha2\_256\_finish()**

```
ATCA_STATUS atcab_hw_sha2_256_finish (
    atca_sha256_ctx_t * ctx,
    uint8_t * digest )
```

Finish SHA-256 digest for a SHA context for performing a hardware SHA-256 operation on a device.

**Parameters**

in	<i>ctx</i>	SHA256 context
out	<i>digest</i>	SHA256 digest is returned here (32 bytes)

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.57 atcab\_hw\_sha2\_256\_init()**

```
ATCA_STATUS atcab_hw_sha2_256_init (
    atca_sha256_ctx_t * ctx )
```

## 18.1 Basic Crypto API methods (atcab\_)

---

Initialize a SHA context for performing a hardware SHA-256 operation on a device. Note that only one SHA operation can be run at a time.

### Parameters

in	ctx	SHA256 context
----	-----	----------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.58 atcab\_hw\_sha2\_256\_update()

```
ATCA_STATUS atcab_hw_sha2_256_update (
    atca_sha256_ctx_t * ctx,
    const uint8_t * data,
    size_t data_size )
```

Add message data to a SHA context for performing a hardware SHA-256 operation on a device.

### Parameters

in	ctx	SHA256 context
in	data	Message data to be added to hash.
in	data_size	Size of data in bytes.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.59 atcab\_idle()

```
ATCA_STATUS atcab_idle (
    void )
```

idle the CryptoAuth device

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.60 atcab\_info()

```
ATCA_STATUS atcab_info (
    uint8_t * revision )
```

Use the Info command to get the device revision (DevRev).



**Parameters**

out	<i>revision</i>	Device revision is returned here (4 bytes).
-----	-----------------	---

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.61 atcab\_info\_base()**

```
ATCA_STATUS atcab_info_base (
    uint8_t mode,
    uint16_t param2,
    uint8_t * out_data )
```

Issues an Info command, which return internal device information and can control GPIO and the persistent latch.

**Parameters**

in	<i>mode</i>	Selects which mode to be used for info command.
in	<i>param2</i>	Selects the particular fields for the mode.
out	<i>out_data</i>	Response from info command (4 bytes). Can be set to NULL if not required.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.62 atcab\_info\_get\_latch()**

```
ATCA_STATUS atcab_info_get_latch (
    bool * state )
```

Use the Info command to get the persistent latch current state for an ATECC608 device.

**Parameters**

out	<i>state</i>	The state is returned here. Set (true) or Cler (false).
-----	--------------	---

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.63 atcab\_info\_set\_latch()

```
ATCA_STATUS atcab_info_set_latch (
    bool state )
```

Use the Info command to set the persistent latch state for an ATECC608 device.

#### Parameters

out	state	Persistent latch state. Set (true) or clear (false).
-----	-------	--

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.64 atcab\_init()

```
ATCA_STATUS atcab_init (
    ATCAIfaceCfg * cfg )
```

Creates a global ATCADevice object used by Basic API.

#### Parameters

in	cfg	Logical interface configuration. Some predefined configurations can be found in <a href="#">atca_cfgs.h</a>
----	-----	---

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.65 atcab\_init\_device()

```
ATCA_STATUS atcab_init_device (
    ATCADevice ca_device )
```

Initialize the global ATCADevice object to point to one of your choosing for use with all the atcab\_ basic API.

**Deprecated** This function is not recommended for use generally. Use of `_ext` is recommended instead. You can use `atcab_init_ext` to obtain an initialized instance and associated it with the global structure - but this shouldn't be a required process except in extremely unusual circumstances.

#### Parameters

in	ca_device	ATCADevice instance to use as the global Basic API crypto device instance
----	-----------	---

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.66 atcab\_init\_ext()**

```
ATCA_STATUS atcab_init_ext (
    ATCADevice * device,
    ATCAIfaceCfg * cfg )
```

Creates and initializes a ATCADevice context.

**Parameters**

out	<i>device</i>	Pointer to the device context pointer
in	<i>cfg</i>	Logical interface configuration. Some predefined configurations can be found in <a href="#">atca_cfgs.h</a>

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.67 atcab\_is\_ca\_device()**

```
bool atcab_is_ca_device (
    ATCADeviceType dev_type )
```

Check whether the device is cryptoauth device.

**Returns**

True if device is cryptoauth device or False.

**18.1.4.68 atcab\_is\_config\_locked()**

```
ATCA_STATUS atcab_is_config_locked (
    bool * is_locked )
```

This function check whether configuration zone is locked or not.

**Parameters**

out	<i>is_locked</i>	Lock state returned here. True if locked.
-----	------------------	---

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.69 atcab\_is\_data\_locked()

```
ATCA_STATUS atcab_is_data_locked (
    bool * is_locked )
```

This function check whether data/setup zone is locked or not.

### Parameters

out	<i>is_locked</i>	Lock state returned here. True if locked.
-----	------------------	---

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.70 atcab\_is\_locked()

```
ATCA_STATUS atcab_is_locked (
    uint8_t zone,
    bool * is_locked )
```

Executes Read command, which reads the configuration zone to see if the specified zone is locked.

### Parameters

in	<i>zone</i>	The zone to query for locked (use LOCK_ZONE_CONFIG or LOCK_ZONE_DATA).
out	<i>is_locked</i>	Lock state returned here. True if locked.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.71 atcab\_is\_slot\_locked()

```
ATCA_STATUS atcab_is_slot_locked (
    uint16_t slot,
    bool * is_locked )
```

This function check whether slot/handle is locked or not.

**Parameters**

in	<i>slot</i>	Slot to query for locked
out	<i>is_locked</i>	Lock state returned here. True if locked.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.72 atcab\_is\_ta\_device()**

```
bool atcab_is_ta_device (
    ATCADeviceType dev_type )
```

Check whether the device is Trust Anchor device.

**Returns**

True if device is Trust Anchor device or False.

**18.1.4.73 atcab\_kdf()**

```
ATCA_STATUS atcab_kdf (
    uint8_t mode,
    uint16_t key_id,
    const uint32_t details,
    const uint8_t * message,
    uint8_t * out_data,
    uint8_t * out_nonce )
```

Executes the KDF command, which derives a new key in PRF, AES, or HKDF modes.

Generally this function combines a source key with an input string and creates a result key/digest/array.

**Parameters**

in	<i>mode</i>	Mode determines KDF algorithm (PRF,AES,HKDF), source key location, and target key locations.
in	<i>key_id</i>	Source and target key slots if locations are in the EEPROM. Source key slot is the LSB and target key slot is the MSB.
in	<i>details</i>	Further information about the computation, depending on the algorithm (4 bytes).
in	<i>message</i>	Input value from system (up to 128 bytes). Actual size of message is 16 bytes for AES algorithm or is encoded in the MSB of the details parameter for other algorithms.
out	<i>out_data</i>	Output of the KDF function is returned here. If the result remains in the device, this can be NULL.
out	<i>out_nonce</i>	If the output is encrypted, a 32 byte random nonce generated by the device is returned here. If output encryption is not used, this can be NULL.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.74 atcab\_lock()

```
ATCA_STATUS atcab_lock (
    uint8_t mode,
    uint16_t summary_crc )
```

The Lock command prevents future modifications of the Configuration and/or Data and OTP zones. If the device is so configured, then this command can be used to lock individual data slots. This command fails if the designated area is already locked.

### Parameters

in	<i>mode</i>	Zone, and/or slot, and summary check (bit 7).
in	<i>summary_crc</i>	CRC of the config or data zones. Ignored for slot locks or when mode bit 7 is set.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.75 atcab\_lock\_config\_zone()

```
ATCA_STATUS atcab_lock_config_zone (
    void )
```

Unconditionally (no CRC required) lock the config zone.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.76 atcab\_lock\_config\_zone\_crc()

```
ATCA_STATUS atcab_lock_config_zone_crc (
    uint16_t summary_crc )
```

Lock the config zone with summary CRC.

The CRC is calculated over the entire config zone contents. 48 bytes for TA100, 88 bytes for ATSHA devices, 128 bytes for ATECC devices. Lock will fail if the provided CRC doesn't match the internally calculated one.

**Parameters**

in	<i>summary_crc</i>	Expected CRC over the config zone.
----	--------------------	------------------------------------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.77 atcab\_lock\_data\_slot()**

```
ATCA_STATUS atcab_lock_data_slot (
    uint16_t slot )
```

Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (KeyConfig.Lockable=1) (for cryptoauth devices) or Lock an individual handle in shared data element on an Trust Anchor device (for Trust Anchor devices).

**Parameters**

in	<i>slot</i>	Slot to be locked in data zone.
----	-------------	---------------------------------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.78 atcab\_lock\_data\_zone()**

```
ATCA_STATUS atcab_lock_data_zone (
    void )
```

Unconditionally (no CRC required) lock the data zone (slots and OTP). for CryptoAuth devices and lock the setup for Trust Anchor device.

ConfigZone must be locked and DataZone must be unlocked for the zone to be successfully locked.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.79 atcab\_lock\_data\_zone\_crc()**

```
ATCA_STATUS atcab_lock_data_zone_crc (
    uint16_t summary_crc )
```

Lock the data zone (slots and OTP) with summary CRC.

The CRC is calculated over the concatenated contents of all the slots and OTP at the end. Private keys (KeyConfig.Private=1) are skipped. Lock will fail if the provided CRC doesn't match the internally calculated one.

## 18.1 Basic Crypto API methods (atcab\_)

---

### Parameters

in	<i>summary_crc</i>	Expected CRC over the data zone.
----	--------------------	----------------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.80 atcab\_mac()

```
ATCA_STATUS atcab_mac (
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * challenge,
    uint8_t * digest )
```

Executes MAC command, which computes a SHA-256 digest of a key stored in the device, a challenge, and other information on the device.

### Parameters

in	<i>mode</i>	Controls which fields within the device are used in the message
in	<i>key_id</i>	Key in the CryptoAuth device to use for the MAC
in	<i>challenge</i>	Challenge message (32 bytes). May be NULL if mode indicates a challenge isn't required.
out	<i>digest</i>	MAC response is returned here (32 bytes).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.81 atcab\_nonce()

```
ATCA_STATUS atcab_nonce (
    const uint8_t * num_in )
```

Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.

### Parameters

in	<i>num_in</i>	Data to be loaded into TempKey (32 bytes).
----	---------------	--



**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.82 atcab\_nonce\_base()**

```
ATCA_STATUS atcab_nonce_base (
    uint8_t mode,
    uint16_t zero,
    const uint8_t * num_in,
    uint8_t * rand_out )
```

Executes Nonce command, which loads a random or fixed nonce/data into the device for use by subsequent commands.

**Parameters**

in	<i>mode</i>	Controls the mechanism of the internal RNG or fixed write.
in	<i>zero</i>	Param2, normally 0, but can be used to indicate a nonce calculation mode (bit 15).
in	<i>num_in</i>	Input value to either be included in the nonce calculation in random modes (20 bytes) or to be written directly (32 bytes or 64 bytes(ATECC608)) in pass-through mode.
out	<i>rand_out</i>	If using a random mode, the internally generated 32-byte random number that was used in the nonce calculation is returned here. Can be NULL if not needed.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.83 atcab\_nonce\_load()**

```
ATCA_STATUS atcab_nonce_load (
    uint8_t target,
    const uint8_t * num_in,
    uint16_t num_in_size )
```

Execute a Nonce command in pass-through mode to load one of the device's internal buffers with a fixed value.

For the ATECC608, available targets are TempKey (32 or 64 bytes), Message Digest Buffer (32 or 64 bytes), or the Alternate Key Buffer (32 bytes). For all other devices, only TempKey (32 bytes) is available.

**Parameters**

in	<i>target</i>	Target device buffer to load. Can be NONCE_MODE_TARGET_TEMPKEY, NONCE_MODE_TARGET_MSGDIGBUF, or NONCE_MODE_TARGET_ALTKEYBUF.
in	<i>num_in</i>	Data to load into the buffer.
in	<i>num_in_size</i>	Size of num_in in bytes. Can be 32 or 64 bytes depending on device and target.

## 18.1 Basic Crypto API methods (atcab\_)

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.84 atcab\_nonce\_rand()

```
ATCA_STATUS atcab_nonce_rand (
    const uint8_t * num_in,
    uint8_t * rand_out )
```

Execute a Nonce command to generate a random nonce combining a host nonce (num\_in) and a device random number.

### Parameters

in	<i>num_in</i>	Host nonce to be combined with the device random number (20 bytes).
out	<i>rand_out</i>	Internally generated 32-byte random number that was used in the nonce/challenge calculation is returned here. Can be NULL if not needed.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.85 atcab\_printbin()

```
ATCA_STATUS atcab_printbin (
    uint8_t * binary,
    size_t bin_len,
    bool add_space )
```

#### 18.1.4.86 atcab\_priv\_write()

```
ATCA_STATUS atcab_priv_write (
    uint16_t key_id,
    const uint8_t priv_key[36],
    uint16_t write_key_id,
    const uint8_t write_key[32],
    const uint8_t num_in[(20)] )
```

Executes PrivWrite command, to write externally generated ECC private keys into the device.

### Parameters

in	<i>key_id</i>	Slot to write the external private key into.
in	<i>priv_key</i>	External private key (36 bytes) to be written. The first 4 bytes should be zero for P256 curve.
in	<i>write_key_id</i>	Write key slot. Ignored if write_key is NULL.
in	<i>write_key</i>	Write key (32 bytes). If NULL, perform an unencrypted PrivWrite, which is only available when the data zone is unlocked.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.87 atcab\_random()**

```
ATCA_STATUS atcab_random (
    uint8_t * rand_out )
```

Executes Random command, which generates a 32 byte random number from the device.

**Parameters**

out	<i>rand_out</i>	32 bytes of random data is returned here.
-----	-----------------	---

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.88 atcab\_random\_ext()**

```
ATCA_STATUS atcab_random_ext (
    ATCADevice device,
    uint8_t * rand_out )
```

Executes Random command, which generates a 32 byte random number from the device.

**Parameters**

in	<i>device</i>	Device context pointer
out	<i>rand_out</i>	32 bytes of random data is returned here.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.89 atcab\_read\_bytes\_zone()**

```
ATCA_STATUS atcab_read_bytes_zone (
    uint8_t zone,
    uint16_t slot,
    size_t offset,
```

## 18.1 Basic Crypto API methods (atcab\_)

---

```
uint8_t * data,  
size_t length )
```

Used to read an arbitrary number of bytes from any zone configured for clear reads.

This function will issue the Read command as many times as is required to read the requested data.

### Parameters

in	zone	Zone to read data from. Option are <a href="#">ATCA_ZONE_CONFIG(0)</a> , <a href="#">ATCA_ZONE_OTP(1)</a> , or <a href="#">ATCA_ZONE_DATA(2)</a> .
in	slot	Slot number to read from if zone is <a href="#">ATCA_ZONE_DATA(2)</a> . Ignored for all other zones.
in	offset	Byte offset within the zone to read from.
out	data	Read data is returned here.
in	length	Number of bytes to read starting from the offset.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.90 atcab\_read\_config\_zone()

```
ATCA_STATUS atcab_read_config_zone (  
    uint8_t * config_data )
```

Executes Read command to read the complete device configuration zone.

### Parameters

out	config_data	Configuration zone data is returned here. 88 bytes for ATSHA devices, 128 bytes for ATECC devices and 48 bytes for Trust Anchor devices.
-----	-------------	--

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.91 atcab\_read\_enc()

```
ATCA_STATUS atcab_read_enc (  
    uint16_t key_id,  
    uint8_t block,  
    uint8_t * data,  
    const uint8_t * enc_key,  
    const uint16_t enc_key_id,  
    const uint8_t num_in[ (20) ] )
```

Executes Read command on a slot configured for encrypted reads and decrypts the data to return it as plaintext.

Data zone must be locked for this command to succeed. Can only read 32 byte blocks.

## Parameters

in	<i>key_id</i>	The slot ID to read from.
in	<i>block</i>	Index of the 32 byte block within the slot to read.
out	<i>data</i>	Decrypted (plaintext) data from the read is returned here (32 bytes).
in	<i>enc_key</i>	32 byte ReadKey for the slot being read.
in	<i>enc_key_id</i>	KeyID of the ReadKey being used.
in	<i>num_in</i>	20 byte host nonce to inject into Nonce calculation

returns ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.92 atcab\_read\_pubkey()

```
ATCA_STATUS atcab_read_pubkey (
    uint16_t slot,
    uint8_t * public_key )
```

Executes Read command to read an ECC P256 public key from a slot configured for clear reads.

This function assumes the public key is stored using the ECC public key format specified in the datasheet.

## Parameters

in	<i>slot</i>	Slot number to read from. Only slots 8 to 15 are large enough for a public key.
out	<i>public_key</i>	Public key is returned here (64 bytes). Format will be the 32 byte X and Y big-endian integers concatenated.

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.93 atcab\_read\_serial\_number()

```
ATCA_STATUS atcab_read_serial_number (
    uint8_t * serial_number )
```

This function returns serial number of the device.

## Parameters

out	<i>serial_number</i>	9 byte serial number is returned here.
-----	----------------------	--

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.94 atcab\_read\_sig()

```
ATCA_STATUS atcab_read_sig (
    uint16_t slot,
    uint8_t * sig )
```

Executes Read command to read a 64 byte ECDSA P256 signature from a slot configured for clear reads.

#### Parameters

in	<i>slot</i>	Slot number to read from. Only slots 8 to 15 are large enough for a signature.
out	<i>sig</i>	Signature will be returned here (64 bytes). Format will be the 32 byte R and S big-endian integers concatenated.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.95 atcab\_read\_zone()

```
ATCA_STATUS atcab_read_zone (
    uint8_t zone,
    uint16_t slot,
    uint8_t block,
    uint8_t offset,
    uint8_t * data,
    uint8_t len )
```

Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.

When reading a slot or OTP, data zone must be locked and the slot configuration must not be secret for a slot to be successfully read.

#### Parameters

in	<i>zone</i>	Zone to be read from device. Options are ATCA_ZONE_CONFIG, ATCA_ZONE_OTP, or ATCA_ZONE_DATA.
in	<i>slot</i>	Slot number for data zone and ignored for other zones.
in	<i>block</i>	32 byte block index within the zone.
in	<i>offset</i>	4 byte work index within the block. Ignored for 32 byte reads.
out	<i>data</i>	Read data is returned here.
in	<i>len</i>	Length of the data to be read. Must be either 4 or 32.

returns ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.96 atcab\_release()**

```
ATCA_STATUS atcab_release (
    void )
```

release (free) the global ATCADevice instance. This must be called in order to release or free up the interface.

**Returns**

Returns ATCA\_SUCCESS .

**18.1.4.97 atcab\_release\_ext()**

```
ATCA_STATUS atcab_release_ext (
    ATCADevice * device )
```

release (free) the an ATCADevice instance.

**Parameters**

in	<i>device</i>	Pointer to the device context pointer
----	---------------	---------------------------------------

**Returns**

Returns ATCA\_SUCCESS .

**18.1.4.98 atcab\_secureboot()**

```
ATCA_STATUS atcab_secureboot (
    uint8_t mode,
    uint16_t param2,
    const uint8_t * digest,
    const uint8_t * signature,
    uint8_t * mac )
```

Executes Secure Boot command, which provides support for secure boot of an external MCU or MPU.

**Parameters**

in	<i>mode</i>	Mode determines what operations the SecureBoot command performs.
in	<i>param2</i>	Not used, must be 0.
in	<i>digest</i>	Digest of the code to be verified (32 bytes).
in	<i>signature</i>	Signature of the code to be verified (64 bytes). Can be NULL when using the FullStore mode.
out	<i>mac</i>	Validating MAC will be returned here (32 bytes). Can be NULL if not required.

## 18.1 Basic Crypto API methods (atcab\_)

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.99 atcab\_secureboot\_mac()

```
ATCA_STATUS atcab_secureboot_mac (
    uint8_t mode,
    const uint8_t * digest,
    const uint8_t * signature,
    const uint8_t * num_in,
    const uint8_t * io_key,
    bool * is_verified )
```

Executes Secure Boot command with encrypted digest and validated MAC response using the IO protection key.

### Parameters

in	<i>mode</i>	Mode determines what operations the SecureBoot command performs.
in	<i>digest</i>	Digest of the code to be verified (32 bytes). This is the plaintext digest (not encrypted).
in	<i>signature</i>	Signature of the code to be verified (64 bytes). Can be NULL when using the FullStore mode.
in	<i>num_in</i>	Host nonce (20 bytes).
in	<i>io_key</i>	IO protection key (32 bytes).
out	<i>is_verified</i>	Verify result is returned here.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.100 atcab\_selftest()

```
ATCA_STATUS atcab_selftest (
    uint8_t mode,
    uint16_t param2,
    uint8_t * result )
```

Executes the SelfTest command, which performs a test of one or more of the cryptographic engines within the ATECC608 chip.

### Parameters

in	<i>mode</i>	Functions to test. Can be a bit field combining any of the following: SELFTEST_MODE_RNG, SELFTEST_MODE_ECDSA_VERIFY, SELFTEST_MODE_ECDSA_SIGN, SELFTEST_MODE_ECDH, SELFTEST_MODE_AES, SELFTEST_MODE_SHA, SELFTEST_MODE_ALL.
in	<i>param2</i>	Currently unused, should be 0.
out	<i>result</i>	Results are returned here as a bit field.



**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.101 atcab\_sha()**

```
ATCA_STATUS atcab_sha (
    uint16_t length,
    const uint8_t * message,
    uint8_t * digest )
```

Use the SHA command to compute a SHA-256 digest.

**Parameters**

in	<i>length</i>	Size of message parameter in bytes.
in	<i>message</i>	Message data to be hashed.
out	<i>digest</i>	Digest is returned here (32 bytes).

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.102 atcab\_sha\_base()**

```
ATCA_STATUS atcab_sha_base (
    uint8_t mode,
    uint16_t length,
    const uint8_t * data_in,
    uint8_t * data_out,
    uint16_t * data_out_size )
```

Executes SHA command, which computes a SHA-256 or HMAC/SHA-256 digest for general purpose use by the host system.

Only the Start(0) and Compute(1) modes are available for ATSHA devices.

**Parameters**

in	<i>mode</i>	SHA command mode Start(0), Update/Compute(1), End(2), Public(3), HMACstart(4), HMACend(5), Read_Context(6), or Write_Context(7). Also message digest target location for the ATECC608.
in	<i>length</i>	Number of bytes in the message parameter or KeySlot for the HMAC key if Mode is HMACstart(4) or Public(3).
in	<i>data_in</i>	Message bytes to be hashed or Write_Context if restoring a context on the ATECC608. Can be NULL if not required by the mode.
out	<i>data_out</i>	Data returned by the command (digest or context).
in, out	<i>data_out_size</i>	As input, the size of the data_out buffer. As output, the number of bytes returned in data_out.

## 18.1 Basic Crypto API methods (atcab\_)

---

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.103 atcab\_sha\_end()

```
ATCA_STATUS atcab_sha_end (
    uint8_t * digest,
    uint16_t length,
    const uint8_t * message )
```

Executes SHA command to complete SHA-256 or HMAC/SHA-256 operation.

### Parameters

out	<i>digest</i>	Digest from SHA-256 or HMAC/SHA-256 will be returned here (32 bytes).
in	<i>length</i>	Length of any remaining data to include in hash. Max 64 bytes.
in	<i>message</i>	Remaining data to include in hash. NULL if length is 0.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.104 atcab\_sha\_hmac()

```
ATCA_STATUS atcab_sha_hmac (
    const uint8_t * data,
    size_t data_size,
    uint16_t key_slot,
    uint8_t * digest,
    uint8_t target )
```

Use the SHA command to compute an HMAC/SHA-256 operation.

### Parameters

in	<i>data</i>	Message data to be hashed.
in	<i>data_size</i>	Size of data in bytes.
in	<i>key_slot</i>	Slot key id to use for the HMAC calculation
out	<i>digest</i>	Digest is returned here (32 bytes).
in	<i>target</i>	Where to save the digest internal to the device. For ATECC608, can be SHA_MODE_TARGET_TEMPKEY, SHA_MODE_TARGET_MSGDIGBUF, or SHA_MODE_TARGET_OUT_ONLY. For all other devices, SHA_MODE_TARGET_TEMPKEY is the only option.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.105 atcab\_sha\_hmac\_finish()**

```
ATCA_STATUS atcab_sha_hmac_finish (
    atca_hmac_sha256_ctx_t * ctx,
    uint8_t * digest,
    uint8_t target )
```

Executes SHA command to complete a HMAC/SHA-256 operation.

**Parameters**

in	<i>ctx</i>	HMAC/SHA-256 context
out	<i>digest</i>	HMAC/SHA-256 result is returned here (32 bytes).
in	<i>target</i>	Where to save the digest internal to the device. For ATECC608, can be SHA_MODE_TARGET_TEMPKEY, SHA_MODE_TARGET_MSGDIGBUF, or SHA_MODE_TARGET_OUT_ONLY. For all other devices, SHA_MODE_TARGET_TEMPKEY is the only option.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.106 atcab\_sha\_hmac\_init()**

```
ATCA_STATUS atcab_sha_hmac_init (
    atca_hmac_sha256_ctx_t * ctx,
    uint16_t key_slot )
```

Executes SHA command to start an HMAC/SHA-256 operation.

**Parameters**

in	<i>ctx</i>	HMAC/SHA-256 context
in	<i>key_slot</i>	Slot key id to use for the HMAC calculation

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.107 atcab\_sha\_hmac\_update()

```
ATCA_STATUS atcab_sha_hmac_update (
    atca_hmac_sha256_ctx_t * ctx,
    const uint8_t * data,
    size_t data_size )
```

Executes SHA command to add an arbitrary amount of message data to a HMAC/SHA-256 operation.

#### Parameters

in	<i>ctx</i>	HMAC/SHA-256 context
in	<i>data</i>	Message data to add
in	<i>data_size</i>	Size of message data in bytes

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.108 atcab\_sha\_read\_context()

```
ATCA_STATUS atcab_sha_read_context (
    uint8_t * context,
    uint16_t * context_size )
```

Executes SHA command to read the SHA-256 context back. Only for ATECC608 with SHA-256 contexts. HMAC not supported.

#### Parameters

out	<i>context</i>	Context data is returned here.
in, out	<i>context_size</i>	As input, the size of the context buffer in bytes. As output, the size of the returned context data.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.109 atcab\_sha\_start()

```
ATCA_STATUS atcab_sha_start (
    void )
```

Executes SHA command to initialize SHA-256 calculation engine.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.110 atcab\_sha\_update()

```
ATCA_STATUS atcab_sha_update (
    const uint8_t * message )
```

Executes SHA command to add 64 bytes of message data to the current context.

##### Parameters

in	<i>message</i>	64 bytes of message data to add to add to operation.
----	----------------	--

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.111 atcab\_sha\_write\_context()

```
ATCA_STATUS atcab_sha_write_context (
    const uint8_t * context,
    uint16_t context_size )
```

Executes SHA command to write (restore) a SHA-256 context into the the device. Only supported for ATECC608 with SHA-256 contexts.

##### Parameters

in	<i>context</i>	Context data to be restored.
in	<i>context_size</i>	Size of the context data in bytes.

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.112 atcab\_sign()

```
ATCA_STATUS atcab_sign (
    uint16_t key_id,
    const uint8_t * msg,
    uint8_t * signature )
```

Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

## 18.1 Basic Crypto API methods (atcab\_)

### Parameters

in	<i>key_id</i>	Slot of the private key to be used to sign the message.
in	<i>msg</i>	32-byte message to be signed. Typically the SHA256 hash of the full message.
out	<i>signature</i>	Signature will be returned here. Format is R and S integers in big-endian format. 64 bytes for P256 curve.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.113 atcab\_sign\_base()

```
ATCA_STATUS atcab_sign_base (
    uint8_t mode,
    uint16_t key_id,
    uint8_t * signature )
```

Executes the Sign command, which generates a signature using the ECDSA algorithm.

### Parameters

in	<i>mode</i>	Mode determines what the source of the message to be signed.
in	<i>key_id</i>	Private key slot used to sign the message.
out	<i>signature</i>	Signature is returned here. Format is R and S integers in big-endian format. 64 bytes for P256 curve.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.114 atcab\_sign\_internal()

```
ATCA_STATUS atcab_sign_internal (
    uint16_t key_id,
    bool is_invalidate,
    bool is_full_sn,
    uint8_t * signature )
```

Executes Sign command to sign an internally generated message.

### Parameters

in	<i>key_id</i>	Slot of the private key to be used to sign the message.
in	<i>is_invalidate</i>	Set to true if the signature will be used with the Verify(Invalid) command. false for all other cases.
in	<i>is_full_sn</i>	Set to true if the message should incorporate the device's full serial number.
out	<i>signature</i>	Signature is returned here. Format is R and S integers in big-endian format. 64 bytes for P256 curve.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.115 atcab\_sleep()**

```
ATCA_STATUS atcab_sleep (
    void )
```

invoke sleep on the CryptoAuth device

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.116 atcab\_updateextra()**

```
ATCA_STATUS atcab_updateextra (
    uint8_t mode,
    uint16_t new_value )
```

Executes UpdateExtra command to update the values of the two extra bytes within the Configuration zone (bytes 84 and 85).

Can also be used to decrement the limited use counter associated with the key in slot NewValue.

**Parameters**

in	<i>mode</i>	Mode determines what operations the UpdateExtra command performs.
in	<i>new_value</i>	Value to be written.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.117 atcab\_verify()**

```
ATCA_STATUS atcab_verify (
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * signature,
    const uint8_t * public_key,
```

## 18.1 Basic Crypto API methods (atcab\_)

```
const uint8_t * other_data,  
uint8_t * mac )
```

Executes the Verify command, which takes an ECDSA [R,S] signature and verifies that it is correctly generated from a given message and public key. In all cases, the signature is an input to the command.

For the Stored, External, and ValidateExternal Modes, the contents of TempKey (or Message Digest Buffer in some cases for the ATECC608) should contain the 32 byte message.

### Parameters

in	<i>mode</i>	Verify command mode and options
in	<i>key_id</i>	Stored mode, the slot containing the public key to be used for the verification. ValidateExternal mode, the slot containing the public key to be validated. External mode, KeyID contains the curve type to be used to Verify the signature. Validate or Invalidate mode, the slot containing the public key to be (in)validated.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>public_key</i>	If mode is External, the public key to be used for verification. X and Y integers in big-endian format. 64 bytes for P256 curve. NULL for all other modes.
in	<i>other_data</i>	If mode is Validate, the bytes used to generate the message for the validation (19 bytes). NULL for all other modes.
out	<i>mac</i>	If mode indicates a validating MAC, then the MAC will be returned here. Can be NULL otherwise.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.118 atcab\_verify\_extern()

```
ATCA_STATUS atcab_verify_extern (  
    const uint8_t * message,  
    const uint8_t * signature,  
    const uint8_t * public_key,  
    bool * is_verified )
```

Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

### Parameters

in	<i>message</i>	32 byte message to be verified. Typically the SHA256 hash of the full message.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>public_key</i>	The public key to be used for verification. X and Y integers in big-endian format. 64 bytes for P256 curve.
out	<i>is_verified</i>	Boolean whether or not the message, signature, public key verified.



**Returns**

ATCA\_SUCCESS on verification success or failure, because the command still completed successfully.

**18.1.4.119 atcab\_verify\_extern\_mac()**

```
ATCA_STATUS atcab_verify_extern_mac (
    const uint8_t * message,
    const uint8_t * signature,
    const uint8_t * public_key,
    const uint8_t * num_in,
    const uint8_t * io_key,
    bool * is_verified )
```

Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. This function is only available on the ATECC608.

**Parameters**

in	<i>message</i>	32 byte message to be verified. Typically the SHA256 hash of the full message.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>public_key</i>	The public key to be used for verification. X and Y integers in big-endian format. 64 bytes for P256 curve.
in	<i>num_in</i>	System nonce (32 byte) used for the verification MAC.
in	<i>io_key</i>	IO protection key for verifying the validation MAC.
out	<i>is_verified</i>	Boolean whether or not the message, signature, public key verified.

**Returns**

ATCA\_SUCCESS on verification success or failure, because the command still completed successfully.

**18.1.4.120 atcab\_verify\_invalidate()**

```
ATCA_STATUS atcab_verify_invalidate (
    uint16_t key_id,
    const uint8_t * signature,
    const uint8_t * other_data,
    bool * is_verified )
```

Executes the Verify command in Invalidate mode which invalidates a previously validated public key stored in a slot.

This command can only be run after GenKey has been used to create a PubKey digest of the public key to be invalidated in TempKey (mode=0x10).

**Parameters**

in	<i>key_id</i>	Slot containing the public key to be invalidated.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>other_data</i>	19 bytes of data used to build the verification message.
out	<i>is_verified</i>	Boolean whether or not the message, signature, validation public key verified.

## 18.1 Basic Crypto API methods (atcab\_)

### Returns

ATCA\_SUCCESS on verification success or failure, because the command still completed successfully.

#### 18.1.4.121 atcab\_verify\_stored()

```
ATCA_STATUS atcab_verify_stored (
    const uint8_t * message,
    const uint8_t * signature,
    uint16_t key_id,
    bool * is_verified )
```

Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

### Parameters

in	<i>message</i>	32 byte message to be verified. Typically the SHA256 hash of the full message.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>key_id</i>	Slot containing the public key to be used in the verification.
out	<i>is_verified</i>	Boolean whether or not the message, signature, public key verified.

### Returns

ATCA\_SUCCESS on verification success or failure, because the command still completed successfully.

#### 18.1.4.122 atcab\_verify\_stored\_mac()

```
ATCA_STATUS atcab_verify_stored_mac (
    const uint8_t * message,
    const uint8_t * signature,
    uint16_t key_id,
    const uint8_t * num_in,
    const uint8_t * io_key,
    bool * is_verified )
```

Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with a public key stored in the device. This function is only available on the ATECC608.

### Parameters

in	<i>message</i>	32 byte message to be verified. Typically the SHA256 hash of the full message.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>key_id</i>	Slot containing the public key to be used in the verification.
in	<i>num_in</i>	System nonce (32 byte) used for the verification MAC.
in	<i>io_key</i>	IO protection key for verifying the validation MAC.
out	<i>is_verified</i>	Boolean whether or not the message, signature, public key verified.

**Returns**

ATCA\_SUCCESS on verification success or failure, because the command still completed successfully.

**18.1.4.123 atcab\_verify\_validate()**

```
ATCA_STATUS atcab_verify_validate (
    uint16_t key_id,
    const uint8_t * signature,
    const uint8_t * other_data,
    bool * is_verified )
```

Executes the Verify command in Validate mode to validate a public key stored in a slot.

This command can only be run after GenKey has been used to create a PubKey digest of the public key to be validated in TempKey (mode=0x10).

**Parameters**

in	<i>key_id</i>	Slot containing the public key to be validated.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>other_data</i>	19 bytes of data used to build the verification message.
out	<i>is_verified</i>	Boolean whether or not the message, signature, validation public key verified.

**Returns**

ATCA\_SUCCESS on verification success or failure, because the command still completed successfully.

**18.1.4.124 atcab\_version()**

```
ATCA_STATUS atcab_version (
    char * ver_str )
```

basic API methods are all prefixed with atcab\_ (CryptoAuthLib Basic) the fundamental premise of the basic API is it is based on a single interface instance and that instance is global, so all basic API commands assume that one global device is the one to operate on.

returns a version string for the CryptoAuthLib release. The format of the version string returned is "yyyymmdd"

**Parameters**

out	<i>ver_str</i>	ptr to space to receive version string
-----	----------------	--

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.125 atcab\_wakeup()

```
ATCA_STATUS atcab_wakeup (
    void )
```

wakeup the CryptoAuth device

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.126 atcab\_write()

```
ATCA_STATUS atcab_write (
    uint8_t zone,
    uint16_t address,
    const uint8_t * value,
    const uint8_t * mac )
```

Executes the Write command, which writes either one four byte word or a 32-byte block to one of the EEPROM zones on the device. Depending upon the value of the WriteConfig byte for this slot, the data may be required to be encrypted by the system prior to being sent to the device. This command cannot be used to write slots configured as ECC private keys.

#### Parameters

in	<i>zone</i>	Zone/Param1 for the write command.
in	<i>address</i>	Address/Param2 for the write command.
in	<i>value</i>	Plain-text data to be written or cipher-text for encrypted writes. 32 or 4 bytes depending on bit 7 in the zone.
in	<i>mac</i>	MAC required for encrypted writes (32 bytes). Set to NULL if not required.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.1.4.127 atcab\_write\_bytes\_zone()

```
ATCA_STATUS atcab_write_bytes_zone (
    uint8_t zone,
    uint16_t slot,
    size_t offset_bytes,
```

```
const uint8_t * data,
size_t length )
```

Executes the Write command, which writes data into the configuration, otp, or data zones with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).

Config zone must be unlocked for writes to that zone. If data zone is unlocked, only 32-byte writes are allowed to slots and OTP and the offset and length must be multiples of 32 or the write will fail.

#### Parameters

in	<i>zone</i>	Zone to write data to: <a href="#">ATCA_ZONE_CONFIG(0)</a> , <a href="#">ATCA_ZONE_OTP(1)</a> , or <a href="#">ATCA_ZONE_DATA(2)</a> .
in	<i>slot</i>	If zone is <a href="#">ATCA_ZONE_DATA(2)</a> , the slot number to write to. Ignored for all other zones.
in	<i>offset_bytes</i>	Byte offset within the zone to write to. Must be a multiple of a word (4 bytes).
in	<i>data</i>	Data to be written.
in	<i>length</i>	Number of bytes to be written. Must be a multiple of a word (4 bytes).

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.128 atcab\_write\_config\_counter()

```
ATCA_STATUS atcab_write_config_counter (
    uint16_t counter_id,
    uint32_t counter_value )
```

Initialize one of the monotonic counters in device with a specific value.

The monotonic counters are stored in the configuration zone using a special format. This encodes a binary count value into the 8 byte encoded value required. Can only be set while the configuration zone is unlocked.

#### Parameters

in	<i>counter_id</i>	Counter to be written.
in	<i>counter_value</i>	Counter value to set.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.129 atcab\_write\_config\_zone()

```
ATCA_STATUS atcab_write_config_zone (
    const uint8_t * config_data )
```

## 18.1 Basic Crypto API methods (atcab\_)

Executes the Write command, which writes the configuration zone.

First 16 bytes are skipped as they are not writable. LockValue and LockConfig are also skipped and can only be changed via the Lock command.

This command may fail if UserExtra and/or Selector bytes have already been set to non-zero values.

### Parameters

in	<i>config_data</i>	Data to the config zone data. This should be 88 bytes for SHA devices and 128 bytes for ECC devices.
----	--------------------	--

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.130 atcab\_write\_enc()

```
ATCA_STATUS atcab_write_enc (
    uint16_t key_id,
    uint8_t block,
    const uint8_t * data,
    const uint8_t * enc_key,
    const uint16_t enc_key_id,
    const uint8_t num_in[ (20) ] )
```

Executes the Write command, which performs an encrypted write of a 32 byte block into given slot.

The function takes clear text bytes and encrypts them for writing over the wire. Data zone must be locked and the slot configuration must be set to encrypted write for the block to be successfully written.

### Parameters

in	<i>key_id</i>	Slot ID to write to.
in	<i>block</i>	Index of the 32 byte block to write in the slot.
in	<i>data</i>	32 bytes of clear text data to be written to the slot
in	<i>enc_key</i>	WriteKey to encrypt with for writing
in	<i>enc_key_id</i>	The KeyID of the WriteKey
in	<i>num_in</i>	20 byte host nonce to inject into Nonce calculation

returns ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.131 atcab\_write\_pubkey()

```
ATCA_STATUS atcab_write_pubkey (
    uint16_t slot,
    const uint8_t * public_key )
```

Uses the write command to write a public key to a slot in the proper format.

## Parameters

in	<i>slot</i>	Slot number to write. Only slots 8 to 15 are large enough to store a public key.
in	<i>public_key</i>	Public key to write into the slot specified. X and Y integers in big-endian format. 64 bytes for P256 curve.

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.132 atcab\_write\_zone()**

```
ATCA_STATUS atcab_write_zone (
    uint8_t zone,
    uint16_t slot,
    uint8_t block,
    uint8_t offset,
    const uint8_t * data,
    uint8_t len )
```

Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.

## Parameters

in	<i>zone</i>	Device zone to write to (0=config, 1=OTP, 2=data).
in	<i>slot</i>	If writing to the data zone, it is the slot to write to, otherwise it should be 0.
in	<i>block</i>	32-byte block to write to.
in	<i>offset</i>	4-byte word within the specified block to write to. If performing a 32-byte write, this should be 0.
in	<i>data</i>	Data to be written.
in	<i>len</i>	Number of bytes to be written. Must be either 4 or 32.

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.1.4.133 calib\_aes\_gcm\_aad\_update()**

```
ATCA_STATUS calib_aes_gcm_aad_update (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * aad,
    uint32_t aad_size )
```

Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.

This can be called multiple times. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function. When there is AAD to include, this should be called before [atcab\\_aes\\_gcm\\_encrypt\\_update\(\)](#) or [atcab\\_aes\\_gcm\\_decrypt\\_update\(\)](#).

### Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES GCM context
in	<i>aad</i>	Additional authenticated data to be added
in	<i>aad_size</i>	Size of aad in bytes

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.134 calib\_aes\_gcm\_decrypt\_finish()

```
ATCA_STATUS calib_aes_gcm_decrypt_finish (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * tag,
    size_t tag_size,
    bool * is_verified )
```

Complete a GCM decrypt operation verifying the authentication tag.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES GCM context structure.
in	<i>tag</i>	Expected authentication tag.
in	<i>tag_size</i>	Size of tag in bytes (12 to 16 bytes).
out	<i>is_verified</i>	Returns whether or not the tag verified.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.135 calib\_aes\_gcm\_decrypt\_update()

```
ATCA_STATUS calib_aes_gcm_decrypt_update (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * ciphertext,
    uint32_t ciphertext_size,
    uint8_t * plaintext )
```

Decrypt data using GCM mode and a key within the ATECC608 device. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function.



## Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES GCM context structure.
in	<i>ciphertext</i>	Ciphertext to be decrypted.
in	<i>ciphertext_size</i>	Size of ciphertext in bytes.
out	<i>plaintext</i>	Decrypted data is returned here.

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 18.1.4.136 calib\_aes\_gcm\_encrypt\_finish()

```
ATCA_STATUS calib_aes_gcm_encrypt_finish (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    uint8_t * tag,
    size_t tag_size )
```

Complete a GCM encrypt operation returning the authentication tag.

## Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES GCM context structure.
out	<i>tag</i>	Authentication tag is returned here.
in	<i>tag_size</i>	Tag size in bytes (12 to 16 bytes).

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 18.1.4.137 calib\_aes\_gcm\_encrypt\_update()

```
ATCA_STATUS calib_aes_gcm_encrypt_update (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * plaintext,
    uint32_t plaintext_size,
    uint8_t * ciphertext )
```

Encrypt data using GCM mode and a key within the ATECC608 device. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function.

## 18.1 Basic Crypto API methods (atcab\_)

---

### Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES GCM context structure.
in	<i>plaintext</i>	Plaintext to be encrypted (16 bytes).
in	<i>plaintext_size</i>	Size of plaintext in bytes.
out	<i>ciphertext</i>	Encrypted data is returned here.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.138 calib\_aes\_gcm\_init()

```
ATCA_STATUS calib_aes_gcm_init (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * iv,
    size_t iv_size )
```

Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES GCM context to be initialized.
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>iv</i>	Initialization vector.
in	<i>iv_size</i>	Size of IV in bytes. Standard is 12 bytes.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.1.4.139 calib\_aes\_gcm\_init\_rand()

```
ATCA_STATUS calib_aes_gcm_init_rand (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block,
    size_t rand_size,
```

```
const uint8_t * free_field,
size_t free_field_size,
uint8_t * iv )
```

Initialize context for AES GCM operation with a IV composed of a random and optional fixed(free) field, which is common when starting an encrypt operation.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES CTR context to be initialized.
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>rand_size</i>	Size of the random field in bytes. Minimum and recommended size is 12 bytes. Max is 32 bytes.
in	<i>free_field</i>	Fixed data to include in the IV after the random field. Can be NULL if not used.
in	<i>free_field_size</i>	Size of the free field in bytes.
out	<i>iv</i>	Initialization vector is returned here. Its size will be <i>rand_size</i> and <i>free_field_size</i> combined.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 18.1.5 Variable Documentation

### 18.1.5.1 `_gDevice`

```
SHARED_LIB_IMPORT ATCADevice _gDevice
```

### 18.1.5.2 `atca_basic_aes_gcm_version`

```
const char* atca_basic_aes_gcm_version
```

### 18.2 Configuration (cfg\_)

Logical device configurations describe the CryptoAuth device type and logical interface.

Logical device configurations describe the CryptoAuth device type and logical interface.

## 18.3 ATCACommand (atca\_)

CryptoAuthLib command builder object, ATCACommand. Member functions for the ATCACommand object.

### Data Structures

- struct [atca\\_command](#)  
*atca\_command is the C object backing ATCACommand.*

### Typedefs

- typedef struct [atca\\_command](#) \* [ATCACommand](#)

### Functions

- [ATCA\\_STATUS](#) [initATCACommand](#) ([ATCADeviceType](#) device\_type, [ATCACommand](#) ca\_cmd)  
*Initializer for ATCACommand.*
- [ATCACommand](#) [newATCACommand](#) ([ATCADeviceType](#) device\_type)  
*constructor for ATCACommand*
- void [deleteATCACommand](#) ([ATCACommand](#) \*ca\_cmd)  
*ATCACommand destructor.*

#### 18.3.1 Detailed Description

CryptoAuthLib command builder object, ATCACommand. Member functions for the ATCACommand object.

#### 18.3.2 Typedef Documentation

##### 18.3.2.1 ATCACommand

```
typedef struct atca\_command* ATCACommand
```

#### 18.3.3 Function Documentation

##### 18.3.3.1 deleteATCACommand()

```
void deleteATCACommand (
    ATCACommand * ca_cmd )
```

ATCACommand destructor.

## 18.3 ATCACCommand (atca\_)

---

### Parameters

in	<i>ca_cmd</i>	instance of a command object
----	---------------	------------------------------

### 18.3.3.2 initATCACCommand()

```
ATCA_STATUS initATCACCommand (
    ATCADeviceType device_type,
    ATCACCommand ca_cmd )
```

Initializer for ATCACCommand.

### Parameters

in	<i>device_type</i>	Specifies which set of commands and execution times should be associated with this command object.
in	<i>ca_cmd</i>	Pre-allocated command structure to initialize.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.3.3.3 newATCACCommand()

```
ATCACCommand newATCACCommand (
    ATCADeviceType device_type )
```

constructor for ATCACCommand

### Parameters

in	<i>device_type</i>	Specifies which set of commands and execution times should be associated with this command object.
----	--------------------	--

### Returns

Initialized object on success. NULL on failure.

## 18.4 ATCADevice (atca\_)

ATCADevice object - composite of command and interface objects.

### Data Structures

- struct [\\_atsha204a\\_config](#)
- struct [\\_atecc508a\\_config](#)
- struct [\\_atecc608\\_config](#)
- struct [atca\\_device](#)

[atca\\_device](#) is the C object backing ATCADevice. See the [atca\\_device.h](#) file for details on the ATCADevice methods

### Macros

- #define [ATCA\\_PACKED](#)
- #define [ATCA\\_AES\\_ENABLE\\_EN\\_SHIFT](#) (0)
- #define [ATCA\\_AES\\_ENABLE\\_EN\\_MASK](#) (0x01u << [ATCA\\_AES\\_ENABLE\\_EN\\_SHIFT](#))
- #define [ATCA\\_I2C\\_ENABLE\\_EN\\_SHIFT](#) (0)
- #define [ATCA\\_I2C\\_ENABLE\\_EN\\_MASK](#) (0x01u << [ATCA\\_I2C\\_ENABLE\\_EN\\_SHIFT](#))
- #define [ATCA\\_COUNTER\\_MATCH\\_EN\\_SHIFT](#) (0)
- #define [ATCA\\_COUNTER\\_MATCH\\_EN\\_MASK](#) (0x01u << [ATCA\\_COUNTER\\_MATCH\\_EN\\_SHIFT](#))
- #define [ATCA\\_COUNTER\\_MATCH\\_KEY\\_SHIFT](#) (4)
- #define [ATCA\\_COUNTER\\_MATCH\\_KEY\\_MASK](#) (0x0Fu << [ATCA\\_COUNTER\\_MATCH\\_KEY\\_SHIFT](#))
- #define [ATCA\\_COUNTER\\_MATCH\\_KEY\(v\)](#) ([ATCA\\_COUNTER\\_MATCH\\_KEY\\_MASK](#) & (v << [ATCA\\_COUNTER\\_MATCH\\_KEY\\_SHIFT](#)))
- #define [ATCA\\_CHIP\\_MODE\\_I2C\\_EXTRA\\_SHIFT](#) (0)
- #define [ATCA\\_CHIP\\_MODE\\_I2C\\_EXTRA\\_MASK](#) (0x01u << [ATCA\\_CHIP\\_MODE\\_I2C\\_EXTRA\\_SHIFT](#))
- #define [ATCA\\_CHIP\\_MODE\\_TTL\\_EN\\_SHIFT](#) (1)
- #define [ATCA\\_CHIP\\_MODE\\_TTL\\_EN\\_MASK](#) (0x01u << [ATCA\\_CHIP\\_MODE\\_TTL\\_EN\\_SHIFT](#))
- #define [ATCA\\_CHIP\\_MODE\\_WDG\\_LONG\\_SHIFT](#) (2)
- #define [ATCA\\_CHIP\\_MODE\\_WDG\\_LONG\\_MASK](#) (0x01u << [ATCA\\_CHIP\\_MODE\\_WDG\\_LONG\\_SHIFT](#))
- #define [ATCA\\_CHIP\\_MODE\\_CLK\\_DIV\\_SHIFT](#) (3)
- #define [ATCA\\_CHIP\\_MODE\\_CLK\\_DIV\\_MASK](#) (0x1Fu << [ATCA\\_CHIP\\_MODE\\_CLK\\_DIV\\_SHIFT](#))
- #define [ATCA\\_CHIP\\_MODE\\_CLK\\_DIV\(v\)](#) ([ATCA\\_CHIP\\_MODE\\_CLK\\_DIV\\_MASK](#) & (v << [ATCA\\_CHIP\\_MODE\\_CLK\\_DIV\\_SHIFT](#)))
- #define [ATCA\\_SLOT\\_CONFIG\\_READKEY\\_SHIFT](#) (0)
- #define [ATCA\\_SLOT\\_CONFIG\\_READKEY\\_MASK](#) (0x0Fu << [ATCA\\_SLOT\\_CONFIG\\_READKEY\\_SHIFT](#))
- #define [ATCA\\_SLOT\\_CONFIG\\_READKEY\(v\)](#) ([ATCA\\_SLOT\\_CONFIG\\_READKEY\\_MASK](#) & (v << [ATCA\\_SLOT\\_CONFIG\\_READKEY\\_SHIFT](#)))
- #define [ATCA\\_SLOT\\_CONFIG\\_NOMAC\\_SHIFT](#) (4)
- #define [ATCA\\_SLOT\\_CONFIG\\_NOMAC\\_MASK](#) (0x01u << [ATCA\\_SLOT\\_CONFIG\\_NOMAC\\_SHIFT](#))
- #define [ATCA\\_SLOT\\_CONFIG\\_LIMITED\\_USE\\_SHIFT](#) (5)
- #define [ATCA\\_SLOT\\_CONFIG\\_LIMITED\\_USE\\_MASK](#) (0x01u << [ATCA\\_SLOT\\_CONFIG\\_LIMITED\\_USE\\_SHIFT](#))
- #define [ATCA\\_SLOT\\_CONFIG\\_ENCRYPTED\\_READ\\_SHIFT](#) (6)
- #define [ATCA\\_SLOT\\_CONFIG\\_ENCRYPTED\\_READ\\_MASK](#) (0x01u << [ATCA\\_SLOT\\_CONFIG\\_ENCRYPTED\\_READ\\_SHIFT](#))
- #define [ATCA\\_SLOT\\_CONFIG\\_IS\\_SECRET\\_SHIFT](#) (7)
- #define [ATCA\\_SLOT\\_CONFIG\\_IS\\_SECRET\\_MASK](#) (0x01u << [ATCA\\_SLOT\\_CONFIG\\_IS\\_SECRET\\_SHIFT](#))
- #define [ATCA\\_SLOT\\_CONFIG\\_WRITE\\_KEY\\_SHIFT](#) (8)
- #define [ATCA\\_SLOT\\_CONFIG\\_WRITE\\_KEY\\_MASK](#) (0x0Fu << [ATCA\\_SLOT\\_CONFIG\\_WRITE\\_KEY\\_SHIFT](#))
- #define [ATCA\\_SLOT\\_CONFIG\\_WRITE\\_KEY\(v\)](#) ([ATCA\\_SLOT\\_CONFIG\\_WRITE\\_KEY\\_MASK](#) & (v << [ATCA\\_SLOT\\_CONFIG\\_WRITE\\_KEY\\_SHIFT](#)))
- #define [ATCA\\_SLOT\\_CONFIG\\_WRITE\\_CONFIG\\_SHIFT](#) (12)
- #define [ATCA\\_SLOT\\_CONFIG\\_WRITE\\_CONFIG\\_MASK](#) (0x0Fu << [ATCA\\_SLOT\\_CONFIG\\_WRITE\\_CONFIG\\_SHIFT](#))

- #define ATCA\_SLOT\_CONFIG\_WRITE\_CONFIG(v) (ATCA\_SLOT\_CONFIG\_WRITE\_CONFIG\_MASK & (v << ATCA\_SLOT\_CONFIG\_WRITE\_CONFIG\_SHIFT))
- #define ATCA\_SLOT\_CONFIG\_EXT\_SIG\_SHIFT (0)
- #define ATCA\_SLOT\_CONFIG\_EXT\_SIG\_MASK (0x01u << ATCA\_SLOT\_CONFIG\_EXT\_SIG\_SHIFT)
- #define ATCA\_SLOT\_CONFIG\_INT\_SIG\_SHIFT (1)
- #define ATCA\_SLOT\_CONFIG\_INT\_SIG\_MASK (0x01u << ATCA\_SLOT\_CONFIG\_INT\_SIG\_SHIFT)
- #define ATCA\_SLOT\_CONFIG\_ECDH\_SHIFT (2)
- #define ATCA\_SLOT\_CONFIG\_ECDH\_MASK (0x01u << ATCA\_SLOT\_CONFIG\_ECDH\_SHIFT)
- #define ATCA\_SLOT\_CONFIG\_WRITE\_ECDH\_SHIFT (3)
- #define ATCA\_SLOT\_CONFIG\_WRITE\_ECDH\_MASK (0x01u << ATCA\_SLOT\_CONFIG\_WRITE\_ECDH\_SHIFT)
- #define ATCA\_SLOT\_CONFIG\_GEN\_KEY\_SHIFT (8)
- #define ATCA\_SLOT\_CONFIG\_GEN\_KEY\_MASK (0x01u << ATCA\_SLOT\_CONFIG\_GEN\_KEY\_SHIFT)
- #define ATCA\_SLOT\_CONFIG\_PRIV\_WRITE\_SHIFT (9)
- #define ATCA\_SLOT\_CONFIG\_PRIV\_WRITE\_MASK (0x01u << ATCA\_SLOT\_CONFIG\_PRIV\_WRITE\_SHIFT)
- #define ATCA\_USE\_LOCK\_ENABLE\_SHIFT (0)
- #define ATCA\_USE\_LOCK\_ENABLE\_MASK (0x0Fu << ATCA\_USE\_LOCK\_ENABLE\_SHIFT)
- #define ATCA\_USE\_LOCK\_KEY\_SHIFT (4)
- #define ATCA\_USE\_LOCK\_KEY\_MASK (0x0Fu << ATCA\_USE\_LOCK\_KEY\_SHIFT)
- #define ATCA\_VOL\_KEY\_PERM\_SLOT\_SHIFT (0)
- #define ATCA\_VOL\_KEY\_PERM\_SLOT\_MASK (0x0Fu << ATCA\_VOL\_KEY\_PERM\_SLOT\_SHIFT)
- #define ATCA\_VOL\_KEY\_PERM\_SLOT(v) (ATCA\_VOL\_KEY\_PERM\_SLOT\_MASK & (v << ATCA\_VOL\_KEY\_PERM\_SLOT\_SHIFT))
- #define ATCA\_VOL\_KEY\_PERM\_EN\_SHIFT (7)
- #define ATCA\_VOL\_KEY\_PERM\_EN\_MASK (0x01u << ATCA\_VOL\_KEY\_PERM\_EN\_SHIFT)
- #define ATCA\_SECURE\_BOOT\_MODE\_SHIFT (0)
- #define ATCA\_SECURE\_BOOT\_MODE\_MASK (0x03u << ATCA\_SECURE\_BOOT\_MODE\_SHIFT)
- #define ATCA\_SECURE\_BOOT\_MODE(v) (ATCA\_SECURE\_BOOT\_MODE\_MASK & (v << ATCA\_SECURE\_BOOT\_MODE\_SHIFT))
- #define ATCA\_SECURE\_BOOT\_PERSIST\_EN\_SHIFT (3)
- #define ATCA\_SECURE\_BOOT\_PERSIST\_EN\_MASK (0x01u << ATCA\_SECURE\_BOOT\_PERSIST\_EN\_SHIFT)
- #define ATCA\_SECURE\_BOOT\_RAND\_NONCE\_SHIFT (4)
- #define ATCA\_SECURE\_BOOT\_RAND\_NONCE\_MASK (0x01u << ATCA\_SECURE\_BOOT\_RAND\_NONCE\_SHIFT)
- #define ATCA\_SECURE\_BOOT\_DIGEST\_SHIFT (8)
- #define ATCA\_SECURE\_BOOT\_DIGEST\_MASK (0x0Fu << ATCA\_SECURE\_BOOT\_DIGEST\_SHIFT)
- #define ATCA\_SECURE\_BOOT\_DIGEST(v) (ATCA\_SECURE\_BOOT\_DIGEST\_MASK & (v << ATCA\_SECURE\_BOOT\_DIGEST\_SHIFT))
- #define ATCA\_SECURE\_BOOT\_PUB\_KEY\_SHIFT (12)
- #define ATCA\_SECURE\_BOOT\_PUB\_KEY\_MASK (0x0Fu << ATCA\_SECURE\_BOOT\_PUB\_KEY\_SHIFT)
- #define ATCA\_SECURE\_BOOT\_PUB\_KEY(v) (ATCA\_SECURE\_BOOT\_PUB\_KEY\_MASK & (v << ATCA\_SECURE\_BOOT\_PUB\_KEY\_SHIFT))
- #define ATCA\_SLOT\_LOCKED(v) ((0x01 << v) & 0xFFFFu)
- #define ATCA\_CHIP\_OPT\_POST\_EN\_SHIFT (0)
- #define ATCA\_CHIP\_OPT\_POST\_EN\_MASK (0x01u << ATCA\_CHIP\_OPT\_POST\_EN\_SHIFT)
- #define ATCA\_CHIP\_OPT\_IO\_PROT\_EN\_SHIFT (1)
- #define ATCA\_CHIP\_OPT\_IO\_PROT\_EN\_MASK (0x01u << ATCA\_CHIP\_OPT\_IO\_PROT\_EN\_SHIFT)
- #define ATCA\_CHIP\_OPT\_KDF\_AES\_EN\_SHIFT (2)
- #define ATCA\_CHIP\_OPT\_KDF\_AES\_EN\_MASK (0x01u << ATCA\_CHIP\_OPT\_KDF\_AES\_EN\_SHIFT)
- #define ATCA\_CHIP\_OPT\_ECDH\_PROT\_SHIFT (8)
- #define ATCA\_CHIP\_OPT\_ECDH\_PROT\_MASK (0x03u << ATCA\_CHIP\_OPT\_ECDH\_PROT\_SHIFT)
- #define ATCA\_CHIP\_OPT\_ECDH\_PROT(v) (ATCA\_CHIP\_OPT\_ECDH\_PROT\_MASK & (v << ATCA\_CHIP\_OPT\_ECDH\_PROT\_SHIFT))
- #define ATCA\_CHIP\_OPT\_KDF\_PROT\_SHIFT (10)
- #define ATCA\_CHIP\_OPT\_KDF\_PROT\_MASK (0x03u << ATCA\_CHIP\_OPT\_KDF\_PROT\_SHIFT)
- #define ATCA\_CHIP\_OPT\_KDF\_PROT(v) (ATCA\_CHIP\_OPT\_KDF\_PROT\_MASK & (v << ATCA\_CHIP\_OPT\_KDF\_PROT\_SHIFT))
- #define ATCA\_CHIP\_OPT\_IO\_PROT\_KEY\_SHIFT (12)
- #define ATCA\_CHIP\_OPT\_IO\_PROT\_KEY\_MASK (0x0Fu << ATCA\_CHIP\_OPT\_IO\_PROT\_KEY\_SHIFT)
- #define ATCA\_CHIP\_OPT\_IO\_PROT\_KEY(v) (ATCA\_CHIP\_OPT\_IO\_PROT\_KEY\_MASK & (v << ATCA\_CHIP\_OPT\_IO\_PROT\_KEY\_SHIFT))
- #define ATCA\_KEY\_CONFIG\_PRIVATE\_SHIFT (0)



- `#define ATCA_KEY_CONFIG_PRIVATE_MASK (0x01u << ATCA_KEY_CONFIG_PRIVATE_SHIFT)`
- `#define ATCA_KEY_CONFIG_PUB_INFO_SHIFT (1)`
- `#define ATCA_KEY_CONFIG_PUB_INFO_MASK (0x01u << ATCA_KEY_CONFIG_PUB_INFO_SHIFT)`
- `#define ATCA_KEY_CONFIG_KEY_TYPE_SHIFT (2)`
- `#define ATCA_KEY_CONFIG_KEY_TYPE_MASK (0x07u << ATCA_KEY_CONFIG_KEY_TYPE_SHIFT)`
- `#define ATCA_KEY_CONFIG_KEY_TYPE(v) (ATCA_KEY_CONFIG_KEY_TYPE_MASK & (v << ATCA_KEY_CONFIG_KEY_TYPE_SHIFT))`
- `#define ATCA_KEY_CONFIG_LOCKABLE_SHIFT (5)`
- `#define ATCA_KEY_CONFIG_LOCKABLE_MASK (0x01u << ATCA_KEY_CONFIG_LOCKABLE_SHIFT)`
- `#define ATCA_KEY_CONFIG_REQ_RANDOM_SHIFT (6)`
- `#define ATCA_KEY_CONFIG_REQ_RANDOM_MASK (0x01u << ATCA_KEY_CONFIG_REQ_RANDOM_SHIFT)`
- `#define ATCA_KEY_CONFIG_REQ_AUTH_SHIFT (7)`
- `#define ATCA_KEY_CONFIG_REQ_AUTH_MASK (0x01u << ATCA_KEY_CONFIG_REQ_AUTH_SHIFT)`
- `#define ATCA_KEY_CONFIG_AUTH_KEY_SHIFT (8)`
- `#define ATCA_KEY_CONFIG_AUTH_KEY_MASK (0x0Fu << ATCA_KEY_CONFIG_AUTH_KEY_SHIFT)`
- `#define ATCA_KEY_CONFIG_AUTH_KEY(v) (ATCA_KEY_CONFIG_AUTH_KEY_MASK & (v << ATCA_KEY_CONFIG_AUTH_KEY_SHIFT))`
- `#define ATCA_KEY_CONFIG_PERSIST_DISABLE_SHIFT (12)`
- `#define ATCA_KEY_CONFIG_PERSIST_DISABLE_MASK (0x01u << ATCA_KEY_CONFIG_PERSIST_DISABLE_SHIFT)`
- `#define ATCA_KEY_CONFIG_RFU_SHIFT (13)`
- `#define ATCA_KEY_CONFIG_RFU_MASK (0x01u << ATCA_KEY_CONFIG_RFU_SHIFT)`
- `#define ATCA_KEY_CONFIG_X509_ID_SHIFT (14)`
- `#define ATCA_KEY_CONFIG_X509_ID_MASK (0x03u << ATCA_KEY_CONFIG_X509_ID_SHIFT)`
- `#define ATCA_KEY_CONFIG_X509_ID(v) (ATCA_KEY_CONFIG_X509_ID_MASK & (v << ATCA_KEY_CONFIG_X509_ID_SHIFT))`

## Typedefs

- `typedef struct _atsha204a_config atsha204a_config_t`
- `typedef struct _atecc508a_config atecc508a_config_t`
- `typedef struct _atecc608_config atecc608_config_t`
- `typedef struct atca_device * ATCADevice`

## Enumerations

- `enum ATCADeviceType {`  
`ATSHA204A = 0, ATECC108A = 1, ATECC508A = 2, ATECC608A = 3,`  
`ATECC608B = 3, ATECC608 = 3, ATSHA206A = 4, TA100 = 0x10,`  
`ATCA_DEV_UNKNOWN = 0x20 }`

*The supported Device type in Cryptoauthlib library.*

## Functions

- `ATCADevice newATCADevice (ATCAIfaceCfg *cfg)`  
*constructor for a Microchip CryptoAuth device*
- `void deleteATCADevice (ATCADevice *ca_dev)`  
*destructor for a device NULLs reference after object is freed*
- `ATCA_STATUS initATCADevice (ATCAIfaceCfg *cfg, ATCADevice ca_dev)`  
*Initializer for an Microchip CryptoAuth device.*
- `ATCACommand atGetCommands (ATCADevice dev)`  
*returns a reference to the ATCACommand object for the device*
- `ATCAIface atGetIFace (ATCADevice dev)`  
*returns a reference to the ATCAIface interface object for the device*
- `ATCA_STATUS releaseATCADevice (ATCADevice ca_dev)`  
*Release any resources associated with the device.*

### 18.4.1 Detailed Description

ATCADevice object - composite of command and interface objects.

### 18.4.2 Macro Definition Documentation

#### 18.4.2.1 ATCA\_AES\_ENABLE\_EN\_MASK

```
#define ATCA_AES_ENABLE_EN_MASK (0x01u << ATCA_AES_ENABLE_EN_SHIFT)
```

#### 18.4.2.2 ATCA\_AES\_ENABLE\_EN\_SHIFT

```
#define ATCA_AES_ENABLE_EN_SHIFT (0)
```

#### 18.4.2.3 ATCA\_CHIP\_MODE\_CLK\_DIV

```
#define ATCA_CHIP_MODE_CLK_DIV(  
    v ) (ATCA_CHIP_MODE_CLK_DIV_MASK & (v << ATCA_CHIP_MODE_CLK_DIV_SHIFT))
```

#### 18.4.2.4 ATCA\_CHIP\_MODE\_CLK\_DIV\_MASK

```
#define ATCA_CHIP_MODE_CLK_DIV_MASK (0x1Fu << ATCA_CHIP_MODE_CLK_DIV_SHIFT)
```

#### 18.4.2.5 ATCA\_CHIP\_MODE\_CLK\_DIV\_SHIFT

```
#define ATCA_CHIP_MODE_CLK_DIV_SHIFT (3)
```

#### 18.4.2.6 ATCA\_CHIP\_MODE\_I2C\_EXTRA\_MASK

```
#define ATCA_CHIP_MODE_I2C_EXTRA_MASK (0x01u << ATCA_CHIP_MODE_I2C_EXTRA_SHIFT)
```

#### 18.4.2.7 ATCA\_CHIP\_MODE\_I2C\_EXTRA\_SHIFT

```
#define ATCA_CHIP_MODE_I2C_EXTRA_SHIFT (0)
```

#### 18.4.2.8 ATCA\_CHIP\_MODE\_TTL\_EN\_MASK

```
#define ATCA_CHIP_MODE_TTL_EN_MASK (0x01u << ATCA_CHIP_MODE_TTL_EN_SHIFT)
```

#### 18.4.2.9 ATCA\_CHIP\_MODE\_TTL\_EN\_SHIFT

```
#define ATCA_CHIP_MODE_TTL_EN_SHIFT (1)
```

#### 18.4.2.10 ATCA\_CHIP\_MODE\_WDG\_LONG\_MASK

```
#define ATCA_CHIP_MODE_WDG_LONG_MASK (0x01u << ATCA_CHIP_MODE_WDG_LONG_SHIFT)
```

#### 18.4.2.11 ATCA\_CHIP\_MODE\_WDG\_LONG\_SHIFT

```
#define ATCA_CHIP_MODE_WDG_LONG_SHIFT (2)
```

#### 18.4.2.12 ATCA\_CHIP\_OPT\_ECDH\_PROT

```
#define ATCA_CHIP_OPT_ECDH_PROT(  
    v ) (ATCA_CHIP_OPT_ECDH_PROT_MASK & (v << ATCA_CHIP_OPT_ECDH_PROT_SHIFT))
```

#### 18.4.2.13 ATCA\_CHIP\_OPT\_ECDH\_PROT\_MASK

```
#define ATCA_CHIP_OPT_ECDH_PROT_MASK (0x03u << ATCA_CHIP_OPT_ECDH_PROT_SHIFT)
```

### 18.4.2.14 ATCA\_CHIP\_OPT\_ECDH\_PROT\_SHIFT

```
#define ATCA_CHIP_OPT_ECDH_PROT_SHIFT (8)
```

### 18.4.2.15 ATCA\_CHIP\_OPT\_IO\_PROT\_EN\_MASK

```
#define ATCA_CHIP_OPT_IO_PROT_EN_MASK (0x01u << ATCA_CHIP_OPT_IO_PROT_EN_SHIFT)
```

### 18.4.2.16 ATCA\_CHIP\_OPT\_IO\_PROT\_EN\_SHIFT

```
#define ATCA_CHIP_OPT_IO_PROT_EN_SHIFT (1)
```

### 18.4.2.17 ATCA\_CHIP\_OPT\_IO\_PROT\_KEY

```
#define ATCA_CHIP_OPT_IO_PROT_KEY(  
    v ) (ATCA_CHIP_OPT_IO_PROT_KEY_MASK & (v << ATCA_CHIP_OPT_IO_PROT_KEY_SHIFT))
```

### 18.4.2.18 ATCA\_CHIP\_OPT\_IO\_PROT\_KEY\_MASK

```
#define ATCA_CHIP_OPT_IO_PROT_KEY_MASK (0x0Fu << ATCA_CHIP_OPT_IO_PROT_KEY_SHIFT)
```

### 18.4.2.19 ATCA\_CHIP\_OPT\_IO\_PROT\_KEY\_SHIFT

```
#define ATCA_CHIP_OPT_IO_PROT_KEY_SHIFT (12)
```

### 18.4.2.20 ATCA\_CHIP\_OPT\_KDF\_AES\_EN\_MASK

```
#define ATCA_CHIP_OPT_KDF_AES_EN_MASK (0x01u << ATCA_CHIP_OPT_KDF_AES_EN_SHIFT)
```

#### 18.4.2.21 ATCA\_CHIP\_OPT\_KDF\_AES\_EN\_SHIFT

```
#define ATCA_CHIP_OPT_KDF_AES_EN_SHIFT (2)
```

#### 18.4.2.22 ATCA\_CHIP\_OPT\_KDF\_PROT

```
#define ATCA_CHIP_OPT_KDF_PROT(  
    v ) (ATCA_CHIP_OPT_KDF_PROT_MASK & (v << ATCA_CHIP_OPT_KDF_PROT_SHIFT))
```

#### 18.4.2.23 ATCA\_CHIP\_OPT\_KDF\_PROT\_MASK

```
#define ATCA_CHIP_OPT_KDF_PROT_MASK (0x03u << ATCA_CHIP_OPT_KDF_PROT_SHIFT)
```

#### 18.4.2.24 ATCA\_CHIP\_OPT\_KDF\_PROT\_SHIFT

```
#define ATCA_CHIP_OPT_KDF_PROT_SHIFT (10)
```

#### 18.4.2.25 ATCA\_CHIP\_OPT\_POST\_EN\_MASK

```
#define ATCA_CHIP_OPT_POST_EN_MASK (0x01u << ATCA_CHIP_OPT_POST_EN_SHIFT)
```

#### 18.4.2.26 ATCA\_CHIP\_OPT\_POST\_EN\_SHIFT

```
#define ATCA_CHIP_OPT_POST_EN_SHIFT (0)
```

#### 18.4.2.27 ATCA\_COUNTER\_MATCH\_EN\_MASK

```
#define ATCA_COUNTER_MATCH_EN_MASK (0x01u << ATCA_COUNTER_MATCH_EN_SHIFT)
```

### 18.4.2.28 ATCA\_COUNTER\_MATCH\_EN\_SHIFT

```
#define ATCA_COUNTER_MATCH_EN_SHIFT (0)
```

### 18.4.2.29 ATCA\_COUNTER\_MATCH\_KEY

```
#define ATCA_COUNTER_MATCH_KEY(  
    v ) (ATCA_COUNTER_MATCH_KEY_MASK & (v << ATCA_COUNTER_MATCH_KEY_SHIFT))
```

### 18.4.2.30 ATCA\_COUNTER\_MATCH\_KEY\_MASK

```
#define ATCA_COUNTER_MATCH_KEY_MASK (0x0Fu << ATCA_COUNTER_MATCH_KEY_SHIFT)
```

### 18.4.2.31 ATCA\_COUNTER\_MATCH\_KEY\_SHIFT

```
#define ATCA_COUNTER_MATCH_KEY_SHIFT (4)
```

### 18.4.2.32 ATCA\_I2C\_ENABLE\_EN\_MASK

```
#define ATCA_I2C_ENABLE_EN_MASK (0x01u << ATCA_I2C_ENABLE_EN_SHIFT)
```

### 18.4.2.33 ATCA\_I2C\_ENABLE\_EN\_SHIFT

```
#define ATCA_I2C_ENABLE_EN_SHIFT (0)
```

### 18.4.2.34 ATCA\_KEY\_CONFIG\_AUTH\_KEY

```
#define ATCA_KEY_CONFIG_AUTH_KEY(  
    v ) (ATCA_KEY_CONFIG_AUTH_KEY_MASK & (v << ATCA_KEY_CONFIG_AUTH_KEY_SHIFT))
```

#### 18.4.2.35 ATCA\_KEY\_CONFIG\_AUTH\_KEY\_MASK

```
#define ATCA_KEY_CONFIG_AUTH_KEY_MASK (0x0Fu << ATCA_KEY_CONFIG_AUTH_KEY_SHIFT)
```

#### 18.4.2.36 ATCA\_KEY\_CONFIG\_AUTH\_KEY\_SHIFT

```
#define ATCA_KEY_CONFIG_AUTH_KEY_SHIFT (8)
```

#### 18.4.2.37 ATCA\_KEY\_CONFIG\_KEY\_TYPE

```
#define ATCA_KEY_CONFIG_KEY_TYPE(  
    v ) (ATCA_KEY_CONFIG_KEY_TYPE_MASK & (v << ATCA_KEY_CONFIG_KEY_TYPE_SHIFT))
```

#### 18.4.2.38 ATCA\_KEY\_CONFIG\_KEY\_TYPE\_MASK

```
#define ATCA_KEY_CONFIG_KEY_TYPE_MASK (0x07u << ATCA_KEY_CONFIG_KEY_TYPE_SHIFT)
```

#### 18.4.2.39 ATCA\_KEY\_CONFIG\_KEY\_TYPE\_SHIFT

```
#define ATCA_KEY_CONFIG_KEY_TYPE_SHIFT (2)
```

#### 18.4.2.40 ATCA\_KEY\_CONFIG\_LOCKABLE\_MASK

```
#define ATCA_KEY_CONFIG_LOCKABLE_MASK (0x01u << ATCA_KEY_CONFIG_LOCKABLE_SHIFT)
```

#### 18.4.2.41 ATCA\_KEY\_CONFIG\_LOCKABLE\_SHIFT

```
#define ATCA_KEY_CONFIG_LOCKABLE_SHIFT (5)
```

### 18.4.2.42 ATCA\_KEY\_CONFIG\_PERSIST\_DISABLE\_MASK

```
#define ATCA_KEY_CONFIG_PERSIST_DISABLE_MASK (0x01u << ATCA_KEY_CONFIG_PERSIST_DISABLE_SHIFT)
```

### 18.4.2.43 ATCA\_KEY\_CONFIG\_PERSIST\_DISABLE\_SHIFT

```
#define ATCA_KEY_CONFIG_PERSIST_DISABLE_SHIFT (12)
```

### 18.4.2.44 ATCA\_KEY\_CONFIG\_PRIVATE\_MASK

```
#define ATCA_KEY_CONFIG_PRIVATE_MASK (0x01u << ATCA_KEY_CONFIG_PRIVATE_SHIFT)
```

### 18.4.2.45 ATCA\_KEY\_CONFIG\_PRIVATE\_SHIFT

```
#define ATCA_KEY_CONFIG_PRIVATE_SHIFT (0)
```

### 18.4.2.46 ATCA\_KEY\_CONFIG\_PUB\_INFO\_MASK

```
#define ATCA_KEY_CONFIG_PUB_INFO_MASK (0x01u << ATCA_KEY_CONFIG_PUB_INFO_SHIFT)
```

### 18.4.2.47 ATCA\_KEY\_CONFIG\_PUB\_INFO\_SHIFT

```
#define ATCA_KEY_CONFIG_PUB_INFO_SHIFT (1)
```

### 18.4.2.48 ATCA\_KEY\_CONFIG\_REQ\_AUTH\_MASK

```
#define ATCA_KEY_CONFIG_REQ_AUTH_MASK (0x01u << ATCA_KEY_CONFIG_REQ_AUTH_SHIFT)
```

### 18.4.2.49 ATCA\_KEY\_CONFIG\_REQ\_AUTH\_SHIFT

```
#define ATCA_KEY_CONFIG_REQ_AUTH_SHIFT (7)
```



#### 18.4.2.50 ATCA\_KEY\_CONFIG\_REQ\_RANDOM\_MASK

```
#define ATCA_KEY_CONFIG_REQ_RANDOM_MASK (0x01u << ATCA_KEY_CONFIG_REQ_RANDOM_SHIFT)
```

#### 18.4.2.51 ATCA\_KEY\_CONFIG\_REQ\_RANDOM\_SHIFT

```
#define ATCA_KEY_CONFIG_REQ_RANDOM_SHIFT (6)
```

#### 18.4.2.52 ATCA\_KEY\_CONFIG\_RFU\_MASK

```
#define ATCA_KEY_CONFIG_RFU_MASK (0x01u << ATCA_KEY_CONFIG_RFU_SHIFT)
```

#### 18.4.2.53 ATCA\_KEY\_CONFIG\_RFU\_SHIFT

```
#define ATCA_KEY_CONFIG_RFU_SHIFT (13)
```

#### 18.4.2.54 ATCA\_KEY\_CONFIG\_X509\_ID

```
#define ATCA_KEY_CONFIG_X509_ID(  
    v ) (ATCA_KEY_CONFIG_X509_ID_MASK & (v << ATCA_KEY_CONFIG_X509_ID_SHIFT))
```

#### 18.4.2.55 ATCA\_KEY\_CONFIG\_X509\_ID\_MASK

```
#define ATCA_KEY_CONFIG_X509_ID_MASK (0x03u << ATCA_KEY_CONFIG_X509_ID_SHIFT)
```

#### 18.4.2.56 ATCA\_KEY\_CONFIG\_X509\_ID\_SHIFT

```
#define ATCA_KEY_CONFIG_X509_ID_SHIFT (14)
```

## 18.4 ATCADevice (atca\_)

---

### 18.4.2.57 ATCA\_PACKED

```
#define ATCA_PACKED
```

### 18.4.2.58 ATCA\_SECURE\_BOOT\_DIGEST

```
#define ATCA_SECURE_BOOT_DIGEST(  
    v ) (ATCA_SECURE_BOOT_DIGEST_MASK & (v << ATCA_SECURE_BOOT_DIGEST_SHIFT))
```

### 18.4.2.59 ATCA\_SECURE\_BOOT\_DIGEST\_MASK

```
#define ATCA_SECURE_BOOT_DIGEST_MASK (0x0Fu << ATCA_SECURE_BOOT_DIGEST_SHIFT)
```

### 18.4.2.60 ATCA\_SECURE\_BOOT\_DIGEST\_SHIFT

```
#define ATCA_SECURE_BOOT_DIGEST_SHIFT (8)
```

### 18.4.2.61 ATCA\_SECURE\_BOOT\_MODE

```
#define ATCA_SECURE_BOOT_MODE(  
    v ) (ATCA_SECURE_BOOT_MODE_MASK & (v << ATCA_SECURE_BOOT_MODE_SHIFT))
```

### 18.4.2.62 ATCA\_SECURE\_BOOT\_MODE\_MASK

```
#define ATCA_SECURE_BOOT_MODE_MASK (0x03u << ATCA_SECURE_BOOT_MODE_SHIFT)
```

### 18.4.2.63 ATCA\_SECURE\_BOOT\_MODE\_SHIFT

```
#define ATCA_SECURE_BOOT_MODE_SHIFT (0)
```

**18.4.2.64 ATCA\_SECURE\_BOOT\_PERSIST\_EN\_MASK**

```
#define ATCA_SECURE_BOOT_PERSIST_EN_MASK (0x01u << ATCA_SECURE_BOOT_PERSIST_EN_SHIFT)
```

**18.4.2.65 ATCA\_SECURE\_BOOT\_PERSIST\_EN\_SHIFT**

```
#define ATCA_SECURE_BOOT_PERSIST_EN_SHIFT (3)
```

**18.4.2.66 ATCA\_SECURE\_BOOT\_PUB\_KEY**

```
#define ATCA_SECURE_BOOT_PUB_KEY(  
    v ) (ATCA_SECURE_BOOT_PUB_KEY_MASK & (v << ATCA_SECURE_BOOT_PUB_KEY_SHIFT))
```

**18.4.2.67 ATCA\_SECURE\_BOOT\_PUB\_KEY\_MASK**

```
#define ATCA_SECURE_BOOT_PUB_KEY_MASK (0x0Fu << ATCA_SECURE_BOOT_PUB_KEY_SHIFT)
```

**18.4.2.68 ATCA\_SECURE\_BOOT\_PUB\_KEY\_SHIFT**

```
#define ATCA_SECURE_BOOT_PUB_KEY_SHIFT (12)
```

**18.4.2.69 ATCA\_SECURE\_BOOT\_RAND\_NONCE\_MASK**

```
#define ATCA_SECURE_BOOT_RAND_NONCE_MASK (0x01u << ATCA_SECURE_BOOT_RAND_NONCE_SHIFT)
```

**18.4.2.70 ATCA\_SECURE\_BOOT\_RAND\_NONCE\_SHIFT**

```
#define ATCA_SECURE_BOOT_RAND_NONCE_SHIFT (4)
```

### 18.4.2.71 ATCA\_SLOT\_CONFIG\_ECDH\_MASK

```
#define ATCA_SLOT_CONFIG_ECDH_MASK (0x01u << ATCA_SLOT_CONFIG_ECDH_SHIFT)
```

### 18.4.2.72 ATCA\_SLOT\_CONFIG\_ECDH\_SHIFT

```
#define ATCA_SLOT_CONFIG_ECDH_SHIFT (2)
```

### 18.4.2.73 ATCA\_SLOT\_CONFIG\_ENCRYPTED\_READ\_MASK

```
#define ATCA_SLOT_CONFIG_ENCRYPTED_READ_MASK (0x01u << ATCA_SLOT_CONFIG_ENCRYPTED_READ_SHIFT)
```

### 18.4.2.74 ATCA\_SLOT\_CONFIG\_ENCRYPTED\_READ\_SHIFT

```
#define ATCA_SLOT_CONFIG_ENCRYPTED_READ_SHIFT (6)
```

### 18.4.2.75 ATCA\_SLOT\_CONFIG\_EXT\_SIG\_MASK

```
#define ATCA_SLOT_CONFIG_EXT_SIG_MASK (0x01u << ATCA_SLOT_CONFIG_EXT_SIG_SHIFT)
```

### 18.4.2.76 ATCA\_SLOT\_CONFIG\_EXT\_SIG\_SHIFT

```
#define ATCA_SLOT_CONFIG_EXT_SIG_SHIFT (0)
```

### 18.4.2.77 ATCA\_SLOT\_CONFIG\_GEN\_KEY\_MASK

```
#define ATCA_SLOT_CONFIG_GEN_KEY_MASK (0x01u << ATCA_SLOT_CONFIG_GEN_KEY_SHIFT)
```

### 18.4.2.78 ATCA\_SLOT\_CONFIG\_GEN\_KEY\_SHIFT

```
#define ATCA_SLOT_CONFIG_GEN_KEY_SHIFT (8)
```

**18.4.2.79 ATCA\_SLOT\_CONFIG\_INT\_SIG\_MASK**

```
#define ATCA_SLOT_CONFIG_INT_SIG_MASK (0x01u << ATCA_SLOT_CONFIG_INT_SIG_SHIFT)
```

**18.4.2.80 ATCA\_SLOT\_CONFIG\_INT\_SIG\_SHIFT**

```
#define ATCA_SLOT_CONFIG_INT_SIG_SHIFT (1)
```

**18.4.2.81 ATCA\_SLOT\_CONFIG\_IS\_SECRET\_MASK**

```
#define ATCA_SLOT_CONFIG_IS_SECRET_MASK (0x01u << ATCA_SLOT_CONFIG_IS_SECRET_SHIFT)
```

**18.4.2.82 ATCA\_SLOT\_CONFIG\_IS\_SECRET\_SHIFT**

```
#define ATCA_SLOT_CONFIG_IS_SECRET_SHIFT (7)
```

**18.4.2.83 ATCA\_SLOT\_CONFIG\_LIMITED\_USE\_MASK**

```
#define ATCA_SLOT_CONFIG_LIMITED_USE_MASK (0x01u << ATCA_SLOT_CONFIG_LIMITED_USE_SHIFT)
```

**18.4.2.84 ATCA\_SLOT\_CONFIG\_LIMITED\_USE\_SHIFT**

```
#define ATCA_SLOT_CONFIG_LIMITED_USE_SHIFT (5)
```

**18.4.2.85 ATCA\_SLOT\_CONFIG\_NOMAC\_MASK**

```
#define ATCA_SLOT_CONFIG_NOMAC_MASK (0x01u << ATCA_SLOT_CONFIG_NOMAC_SHIFT)
```

**18.4.2.86 ATCA\_SLOT\_CONFIG\_NOMAC\_SHIFT**

```
#define ATCA_SLOT_CONFIG_NOMAC_SHIFT (4)
```

## 18.4 ATCADevice (atca\_)

---

### 18.4.2.87 ATCA\_SLOT\_CONFIG\_PRIV\_WRITE\_MASK

```
#define ATCA_SLOT_CONFIG_PRIV_WRITE_MASK (0x01u << ATCA_SLOT_CONFIG_PRIV_WRITE_SHIFT)
```

### 18.4.2.88 ATCA\_SLOT\_CONFIG\_PRIV\_WRITE\_SHIFT

```
#define ATCA_SLOT_CONFIG_PRIV_WRITE_SHIFT (9)
```

### 18.4.2.89 ATCA\_SLOT\_CONFIG\_READKEY

```
#define ATCA_SLOT_CONFIG_READKEY(  
    v ) (ATCA_SLOT_CONFIG_READKEY_MASK & (v << ATCA_SLOT_CONFIG_READKEY_SHIFT))
```

### 18.4.2.90 ATCA\_SLOT\_CONFIG\_READKEY\_MASK

```
#define ATCA_SLOT_CONFIG_READKEY_MASK (0x0Fu << ATCA_SLOT_CONFIG_READKEY_SHIFT)
```

### 18.4.2.91 ATCA\_SLOT\_CONFIG\_READKEY\_SHIFT

```
#define ATCA_SLOT_CONFIG_READKEY_SHIFT (0)
```

### 18.4.2.92 ATCA\_SLOT\_CONFIG\_WRITE\_CONFIG

```
#define ATCA_SLOT_CONFIG_WRITE_CONFIG(  
    v ) (ATCA_SLOT_CONFIG_WRITE_CONFIG_MASK & (v << ATCA_SLOT_CONFIG_WRITE_CONFIG_SHIFT))
```

### 18.4.2.93 ATCA\_SLOT\_CONFIG\_WRITE\_CONFIG\_MASK

```
#define ATCA_SLOT_CONFIG_WRITE_CONFIG_MASK (0x0Fu << ATCA_SLOT_CONFIG_WRITE_CONFIG_SHIFT)
```

**18.4.2.94 ATCA\_SLOT\_CONFIG\_WRITE\_CONFIG\_SHIFT**

```
#define ATCA_SLOT_CONFIG_WRITE_CONFIG_SHIFT (12)
```

**18.4.2.95 ATCA\_SLOT\_CONFIG\_WRITE\_ECDH\_MASK**

```
#define ATCA_SLOT_CONFIG_WRITE_ECDH_MASK (0x01u << ATCA_SLOT_CONFIG_WRITE_ECDH_SHIFT)
```

**18.4.2.96 ATCA\_SLOT\_CONFIG\_WRITE\_ECDH\_SHIFT**

```
#define ATCA_SLOT_CONFIG_WRITE_ECDH_SHIFT (3)
```

**18.4.2.97 ATCA\_SLOT\_CONFIG\_WRITE\_KEY**

```
#define ATCA_SLOT_CONFIG_WRITE_KEY(  
    v ) (ATCA_SLOT_CONFIG_WRITE_KEY_MASK & (v << ATCA_SLOT_CONFIG_WRITE_KEY_SHIFT))
```

**18.4.2.98 ATCA\_SLOT\_CONFIG\_WRITE\_KEY\_MASK**

```
#define ATCA_SLOT_CONFIG_WRITE_KEY_MASK (0x0Fu << ATCA_SLOT_CONFIG_WRITE_KEY_SHIFT)
```

**18.4.2.99 ATCA\_SLOT\_CONFIG\_WRITE\_KEY\_SHIFT**

```
#define ATCA_SLOT_CONFIG_WRITE_KEY_SHIFT (8)
```

**18.4.2.100 ATCA\_SLOT\_LOCKED**

```
#define ATCA_SLOT_LOCKED(  
    v ) ((0x01 << v) & 0xFFFFu)
```

## 18.4 ATCADevice (atca\_)

---

### 18.4.2.101 ATCA\_USE\_LOCK\_ENABLE\_MASK

```
#define ATCA_USE_LOCK_ENABLE_MASK (0x0Fu << ATCA_USE_LOCK_ENABLE_SHIFT)
```

### 18.4.2.102 ATCA\_USE\_LOCK\_ENABLE\_SHIFT

```
#define ATCA_USE_LOCK_ENABLE_SHIFT (0)
```

### 18.4.2.103 ATCA\_USE\_LOCK\_KEY\_MASK

```
#define ATCA_USE_LOCK_KEY_MASK (0x0Fu << ATCA_USE_LOCK_KEY_SHIFT)
```

### 18.4.2.104 ATCA\_USE\_LOCK\_KEY\_SHIFT

```
#define ATCA_USE_LOCK_KEY_SHIFT (4)
```

### 18.4.2.105 ATCA\_VOL\_KEY\_PERM\_EN\_MASK

```
#define ATCA_VOL_KEY_PERM_EN_MASK (0x01u << ATCA_VOL_KEY_PERM_EN_SHIFT)
```

### 18.4.2.106 ATCA\_VOL\_KEY\_PERM\_EN\_SHIFT

```
#define ATCA_VOL_KEY_PERM_EN_SHIFT (7)
```

### 18.4.2.107 ATCA\_VOL\_KEY\_PERM\_SLOT

```
#define ATCA_VOL_KEY_PERM_SLOT(  
    v ) (ATCA_VOL_KEY_PERM_SLOT_MASK & (v << ATCA_VOL_KEY_PERM_SLOT_SHIFT))
```



#### 18.4.2.108 ATCA\_VOL\_KEY\_PERM\_SLOT\_MASK

```
#define ATCA_VOL_KEY_PERM_SLOT_MASK (0x0Fu << ATCA_VOL_KEY_PERM_SLOT_SHIFT)
```

#### 18.4.2.109 ATCA\_VOL\_KEY\_PERM\_SLOT\_SHIFT

```
#define ATCA_VOL_KEY_PERM_SLOT_SHIFT (0)
```

### 18.4.3 Typedef Documentation

#### 18.4.3.1 ATCADevice

```
typedef struct atca_device* ATCADevice
```

#### 18.4.3.2 atecc508a\_config\_t

```
typedef struct _atecc508a_config atecc508a_config_t
```

#### 18.4.3.3 atecc608\_config\_t

```
typedef struct _atecc608_config atecc608_config_t
```

#### 18.4.3.4 atsha204a\_config\_t

```
typedef struct _atsha204a_config atsha204a_config_t
```

### 18.4.4 Enumeration Type Documentation

#### 18.4.4.1 ATCADeviceType

```
enum ATCADeviceType
```

The supported Device type in Cryptoauthlib library.

### Enumerator

ATSHA204A	
ATECC108A	
ATECC508A	
ATECC608A	
ATECC608B	
ATECC608	
ATSHA206A	
TA100	
ATCA_DEV_UNKNOWN	

## 18.4.5 Function Documentation

### 18.4.5.1 atGetCommands()

```
ATCACommand atGetCommands (  
    ATCADevice dev )
```

returns a reference to the ATCACommand object for the device

#### Parameters

in	<i>dev</i>	reference to a device
----	------------	-----------------------

#### Returns

reference to the ATCACommand object for the device

### 18.4.5.2 atGetIFace()

```
ATCAIface atGetIFace (  
    ATCADevice dev )
```

returns a reference to the ATCAIface interface object for the device

#### Parameters

in	<i>dev</i>	reference to a device
----	------------	-----------------------

**Returns**

reference to the ATCAIface object for the device

**18.4.5.3 deleteATCADevice()**

```
void deleteATCADevice (
    ATCADevice * ca_dev )
```

destructor for a device NULLs reference after object is freed

**Parameters**

in	<i>ca_dev</i>	pointer to a reference to a device
----	---------------	------------------------------------

**18.4.5.4 initATCADevice()**

```
ATCA_STATUS initATCADevice (
    ATCAIfaceCfg * cfg,
    ATCADevice ca_dev )
```

Initializer for an Microchip CryptoAuth device.

**Parameters**

in	<i>cfg</i>	pointer to an interface configuration object
in, out	<i>ca_dev</i>	As input, pre-allocated structure to be initialized. mCommands and mIface members should point to existing structures to be initialized.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.4.5.5 newATCADevice()**

```
ATCADevice newATCADevice (
    ATCAIfaceCfg * cfg )
```

constructor for a Microchip CryptoAuth device

**Parameters**

in	<i>cfg</i>	Interface configuration object
----	------------	--------------------------------

## 18.4 ATCADevice (atca\_)

---

### Returns

Reference to a new ATCADevice on success. NULL on failure.

### 18.4.5.6 releaseATCADevice()

```
ATCA_STATUS releaseATCADevice (  
    ATCADevice ca_dev )
```

Release any resources associated with the device.

### Parameters

in	<i>ca_dev</i>	Device to release
----	---------------	-------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 18.5 ATCAIface (atca\_)

Abstract interface to all CryptoAuth device types. This interface connects to the HAL implementation and abstracts the physical details of the device communication from all the upper layers of CryptoAuthLib.

### Data Structures

- struct [ATCAIfaceCfg](#)
  - struct [atca\\_iface](#)
- [atca\\_iface](#) is the C object backing ATCAIface. See the [atca\\_iface.h](#) file for details on the ATCAIface methods*

### Typedefs

- typedef struct [atca\\_iface](#) \* [ATCAIface](#)

### Enumerations

- enum [ATCAIfaceType](#) {  
[ATCA\\_I2C\\_IFACE](#), [ATCA\\_SWI\\_IFACE](#), [ATCA\\_UART\\_IFACE](#), [ATCA\\_SPI\\_IFACE](#),  
[ATCA\\_HID\\_IFACE](#), [ATCA\\_CUSTOM\\_IFACE](#), [ATCA\\_UNKNOWN\\_IFACE](#) }
- enum [ATCAKitType](#) {  
[ATCA\\_KIT\\_AUTO\\_IFACE](#), [ATCA\\_KIT\\_I2C\\_IFACE](#), [ATCA\\_KIT\\_SWI\\_IFACE](#), [ATCA\\_KIT\\_SPI\\_IFACE](#),  
[ATCA\\_KIT\\_UNKNOWN\\_IFACE](#) }

### Functions

- [ATCA\\_STATUS](#) [initATCAIface](#) ([ATCAIfaceCfg](#) \*cfg, [ATCAIface](#) ca\_iface)  
*Initializer for ATCAIface objects.*
- [ATCAIface](#) [newATCAIface](#) ([ATCAIfaceCfg](#) \*cfg)  
*Constructor for ATCAIface objects.*
- [ATCA\\_STATUS](#) [atinit](#) ([ATCAIface](#) ca\_iface)  
*Performs the HAL initialization by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_init\(\)](#) function should be called instead.*
- [ATCA\\_STATUS](#) [atsend](#) ([ATCAIface](#) ca\_iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*Sends the data to the device by calling intermediate HAL wrapper function.*
- [ATCA\\_STATUS](#) [atreceive](#) ([ATCAIface](#) ca\_iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)  
*Receives data from the device by calling intermediate HAL wrapper function.*
- [ATCA\\_STATUS](#) [atwake](#) ([ATCAIface](#) ca\_iface)  
*Wakes up the device by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_wakeup\(\)](#) function should be used instead.*
- [ATCA\\_STATUS](#) [atidle](#) ([ATCAIface](#) ca\_iface)  
*Puts the device into idle state by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_idle\(\)](#) function should be used instead.*
- [ATCA\\_STATUS](#) [atsleep](#) ([ATCAIface](#) ca\_iface)  
*Puts the device into sleep state by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_sleep\(\)](#) function should be used instead.*
- [ATCAIfaceCfg](#) \* [atgetifacecfg](#) ([ATCAIface](#) ca\_iface)  
*Returns the logical interface configuration for the device.*
- void \* [atgetifacehaldat](#) ([ATCAIface](#) ca\_iface)  
*Returns the HAL data pointer for the device.*
- [ATCA\\_STATUS](#) [releaseATCAIface](#) ([ATCAIface](#) ca\_iface)  
*Instruct the HAL driver to release any resources associated with this interface.*
- void [deleteATCAIface](#) ([ATCAIface](#) \*ca\_iface)  
*Instruct the HAL driver to release any resources associated with this interface, then delete the object.*
- [ATCA\\_STATUS](#) [atpostinit](#) ([ATCAIface](#) ca\_iface)

### 18.5.1 Detailed Description

Abstract interface to all CryptoAuth device types. This interface connects to the HAL implementation and abstracts the physical details of the device communication from all the upper layers of CryptoAuthLib.

### 18.5.2 Typedef Documentation

#### 18.5.2.1 ATCAIface

```
typedef struct atca_iface* ATCAIface
```

### 18.5.3 Enumeration Type Documentation

#### 18.5.3.1 ATCAIfaceType

```
enum ATCAIfaceType
```

##### Enumerator

ATCA_I2C_IFACE	
ATCA_SWI_IFACE	
ATCA_UART_IFACE	
ATCA_SPI_IFACE	
ATCA_HID_IFACE	
ATCA_CUSTOM_IFACE	
ATCA_UNKNOWN_IFACE	

#### 18.5.3.2 ATCAKitType

```
enum ATCAKitType
```

##### Enumerator

ATCA_KIT_AUTO_IFACE	
ATCA_KIT_I2C_IFACE	
ATCA_KIT_SWI_IFACE	
ATCA_KIT_SPI_IFACE	
ATCA_KIT_UNKNOWN_IFACE	

## 18.5.4 Function Documentation

### 18.5.4.1 atgetifacecfg()

```
ATCAIfaceCfg * atgetifacecfg (
    ATCAIface ca_iface )
```

Returns the logical interface configuration for the device.

#### Parameters

in	<i>ca_iface</i>	Device interface.
----	-----------------	-------------------

#### Returns

Logical interface configuration.

### 18.5.4.2 atgetifacehaldat()

```
void * atgetifacehaldat (
    ATCAIface ca_iface )
```

Returns the HAL data pointer for the device.

#### Parameters

in	<i>ca_iface</i>	Device interface.
----	-----------------	-------------------

#### Returns

HAL data pointer.

### 18.5.4.3 atidle()

```
ATCA_STATUS atidle (
    ATCAIface ca_iface )
```

Puts the device into idle state by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_idle\(\)](#) function should be used instead.

## 18.5 ATCAIface (atca\_)

---

### Parameters

in	<i>ca_iface</i>	Device to interact with.
----	-----------------	--------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.5.4.4 atinit()

```
ATCA_STATUS atinit (
    ATCAIface ca_iface )
```

Performs the HAL initialization by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_init\(\)](#) function should be called instead.

### Parameters

in	<i>ca_iface</i>	Device to interact with.
----	-----------------	--------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.5.4.5 atpostinit()

```
ATCA_STATUS atpostinit (
    ATCAIface ca_iface )
```

#### 18.5.4.6 atreceive()

```
ATCA_STATUS atreceive (
    ATCAIface ca_iface,
    uint8_t word_address,
    uint8_t * rxdata,
    uint16_t * rxlength )
```

Receives data from the device by calling intermediate HAL wrapper function.

### Parameters

in	<i>ca_iface</i>	Device to interact with.
in	<i>word_address</i>	device transaction type
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.



**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.5.4.7 atsend()**

```
ATCA_STATUS atsend (
    ATCAIface ca_iface,
    uint8_t word_address,
    uint8_t * txdata,
    int txlength )
```

Sends the data to the device by calling intermediate HAL wrapper function.

**Parameters**

in	<i>ca_iface</i>	Device to interact with.
in	<i>word_address</i>	device transaction type
in	<i>txdata</i>	Data to be transmitted to the device.
in	<i>txlength</i>	Number of bytes to be transmitted to the device.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.5.4.8 atsleep()**

```
ATCA_STATUS atsleep (
    ATCAIface ca_iface )
```

Puts the device into sleep state by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_sleep\(\)](#) function should be used instead.

**Parameters**

in	<i>ca_iface</i>	Device to interact with.
----	-----------------	--------------------------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

## 18.5 ATCAIface (atca\_)

---

### 18.5.4.9 atwake()

```
ATCA_STATUS atwake (
    ATCAIface ca_iface )
```

Wakes up the device by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_wakeup\(\)](#) function should be used instead.

#### Parameters

in	<i>ca_iface</i>	Device to interact with.
----	-----------------	--------------------------

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.5.4.10 deleteATCAIface()

```
void deleteATCAIface (
    ATCAIface * ca_iface )
```

Instruct the HAL driver to release any resources associated with this interface, then delete the object.

#### Parameters

in	<i>ca_iface</i>	Device interface.
----	-----------------	-------------------

### 18.5.4.11 initATCAIface()

```
ATCA_STATUS initATCAIface (
    ATCAIfaceCfg * cfg,
    ATCAIface ca_iface )
```

Initializer for ATCAIface objects.

#### Parameters

in	<i>cfg</i>	Logical configuration for the interface
in	<i>ca_iface</i>	Interface structure to initialize.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.5.4.12 newATCAIface()

```
ATCAIface newATCAIface (
    ATCAIfaceCfg * cfg )
```

Constructor for ATCAIface objects.

##### Parameters

in	<i>cfg</i>	Logical configuration for the interface
----	------------	---

##### Returns

New interface instance on success. NULL on failure.

#### 18.5.4.13 releaseATCAIface()

```
ATCA_STATUS releaseATCAIface (
    ATCAIface ca_iface )
```

Instruct the HAL driver to release any resources associated with this interface.

##### Parameters

in	<i>ca_iface</i>	Device interface.
----	-----------------	-------------------

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 18.6 Certificate manipulation methods (atcacert\_)

These methods provide convenient ways to perform certification I/O with CryptoAuth chips and perform certificate manipulation in memory.

### Data Structures

- struct [atcacert\\_tm\\_utc\\_s](#)
- struct [atcacert\\_device\\_loc\\_s](#)
- struct [atcacert\\_cert\\_loc\\_s](#)
- struct [atcacert\\_cert\\_element\\_s](#)
- struct [atcacert\\_def\\_s](#)
- struct [atcacert\\_build\\_state\\_s](#)

### Macros

- #define [FALSE](#) (0)
- #define [TRUE](#) (1)
- #define [ATCACERT\\_E\\_SUCCESS](#) 0  
*Operation completed successfully.*
- #define [ATCACERT\\_E\\_ERROR](#) 1  
*General error.*
- #define [ATCACERT\\_E\\_BAD\\_PARAMS](#) 2  
*Invalid/bad parameter passed to function.*
- #define [ATCACERT\\_E\\_BUFFER\\_TOO\\_SMALL](#) 3  
*Supplied buffer for output is too small to hold the result.*
- #define [ATCACERT\\_E\\_DECODING\\_ERROR](#) 4  
*Data being decoded/parsed has an invalid format.*
- #define [ATCACERT\\_E\\_INVALID\\_DATE](#) 5  
*Date is invalid.*
- #define [ATCACERT\\_E\\_UNIMPLEMENTED](#) 6  
*Function is unimplemented for the current configuration.*
- #define [ATCACERT\\_E\\_UNEXPECTED\\_ELEM\\_SIZE](#) 7  
*A certificate element size was not what was expected.*
- #define [ATCACERT\\_E\\_ELEM\\_MISSING](#) 8  
*The certificate element isn't defined for the certificate definition.*
- #define [ATCACERT\\_E\\_ELEM\\_OUT\\_OF\\_BOUNDS](#) 9  
*Certificate element is out of bounds for the given certificate.*
- #define [ATCACERT\\_E\\_BAD\\_CERT](#) 10  
*Certificate structure is bad in some way.*
- #define [ATCACERT\\_E\\_WRONG\\_CERT\\_DEF](#) 11
- #define [ATCACERT\\_E\\_VERIFY\\_FAILED](#) 12  
*Certificate or challenge/response verification failed.*
- #define [ATCACERT\\_E\\_INVALID\\_TRANSFORM](#) 13  
*Invalid transform passed to function.*
- #define [DATEFMT\\_ISO8601\\_SEP\\_SIZE](#) (20)
- #define [DATEFMT\\_RFC5280\\_UTC\\_SIZE](#) (13)
- #define [DATEFMT\\_POSIX\\_UINT32\\_BE\\_SIZE](#) (4)
- #define [DATEFMT\\_POSIX\\_UINT32\\_LE\\_SIZE](#) (4)
- #define [DATEFMT\\_RFC5280\\_GEN\\_SIZE](#) (15)
- #define [DATEFMT\\_MAX\\_SIZE](#) [DATEFMT\\_ISO8601\\_SEP\\_SIZE](#)
- #define [ATCACERT\\_DATE\\_FORMAT\\_SIZES\\_COUNT](#) 5
- #define [ATCA\\_PACKED](#)

## Typedefs

- typedef struct [atcacert\\_tm\\_utc\\_s](#) [atcacert\\_tm\\_utc\\_t](#)
- typedef enum [atcacert\\_date\\_format\\_e](#) [atcacert\\_date\\_format\\_t](#)
- typedef enum [atcacert\\_cert\\_type\\_e](#) [atcacert\\_cert\\_type\\_t](#)
- typedef enum [atcacert\\_cert\\_sn\\_src\\_e](#) [atcacert\\_cert\\_sn\\_src\\_t](#)
- typedef enum [atcacert\\_device\\_zone\\_e](#) [atcacert\\_device\\_zone\\_t](#)
- typedef enum [atcacert\\_transform\\_e](#) [atcacert\\_transform\\_t](#)
- *How to transform the data from the device to the certificate.*
- typedef enum [atcacert\\_std\\_cert\\_element\\_e](#) [atcacert\\_std\\_cert\\_element\\_t](#)
- typedef struct [atcacert\\_device\\_loc\\_s](#) [atcacert\\_device\\_loc\\_t](#)
- typedef struct [atcacert\\_cert\\_loc\\_s](#) [atcacert\\_cert\\_loc\\_t](#)
- typedef struct [atcacert\\_cert\\_element\\_s](#) [atcacert\\_cert\\_element\\_t](#)
- typedef struct [atcacert\\_def\\_s](#) [atcacert\\_def\\_t](#)
- typedef struct [atcacert\\_build\\_state\\_s](#) [atcacert\\_build\\_state\\_t](#)

## Enumerations

- enum [atcacert\\_date\\_format\\_e](#) {  
DATEFMT\_ISO8601\_SEP, DATEFMT\_RFC5280\_UTC, DATEFMT\_POSIX\_UINT32\_BE, DATEFMT\_POSIX\_UINT32\_LE,  
DATEFMT\_RFC5280\_GEN }
- enum [atcacert\\_cert\\_type\\_e](#) { CERTTYPE\_X509, CERTTYPE\_CUSTOM }
- enum [atcacert\\_cert\\_sn\\_src\\_e](#) {  
SNSRC\_STORED = 0x0, SNSRC\_STORED\_DYNAMIC = 0x7, SNSRC\_DEVICE\_SN = 0x8, SNSRC\_SIGNER\_ID  
= 0x9,  
SNSRC\_PUB\_KEY\_HASH = 0xA, SNSRC\_DEVICE\_SN\_HASH = 0xB, SNSRC\_PUB\_KEY\_HASH\_POS =  
0xC, SNSRC\_DEVICE\_SN\_HASH\_POS = 0xD,  
SNSRC\_PUB\_KEY\_HASH\_RAW = 0xE, SNSRC\_DEVICE\_SN\_HASH\_RAW = 0xF }
- enum [atcacert\\_device\\_zone\\_e](#) { DEVZONE\_CONFIG = 0x00, DEVZONE\_OTP = 0x01, DEVZONE\_DATA =  
0x02, DEVZONE\_NONE = 0x07 }
- enum [atcacert\\_transform\\_e](#) {  
TF\_NONE, TF\_REVERSE, TF\_BIN2HEX\_UC, TF\_BIN2HEX\_LC,  
TF\_HEX2BIN\_UC, TF\_HEX2BIN\_LC, TF\_BIN2HEX\_SPACE\_UC, TF\_BIN2HEX\_SPACE\_LC,  
TF\_HEX2BIN\_SPACE\_UC, TF\_HEX2BIN\_SPACE\_LC }
- *How to transform the data from the device to the certificate.*
- enum [atcacert\\_std\\_cert\\_element\\_e](#) {  
STDCERT\_PUBLIC\_KEY, STDCERT\_SIGNATURE, STDCERT\_ISSUE\_DATE, STDCERT\_EXPIRE\_DATE,  
STDCERT\_SIGNER\_ID, STDCERT\_CERT\_SN, STDCERT\_AUTH\_KEY\_ID, STDCERT\_SUBJ\_KEY\_ID,  
STDCERT\_NUM\_ELEMENTS }

## Functions

- int [atcacert\\_read\\_device\\_loc](#) (const [atcacert\\_device\\_loc\\_t](#) \*device\_loc, uint8\_t \*data)  
*Read the data from a device location.*
- int [atcacert\\_read\\_cert](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t ca\_public\_key[64], uint8\_t \*cert, size\_t  
\*cert\_size)  
*Reads the certificate specified by the certificate definition from the ATECC508A device.*
- int [atcacert\\_write\\_cert](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size)  
*Take a full certificate and write it to the ATECC508A device according to the certificate definition.*
- int [atcacert\\_create\\_csr](#) (const [atcacert\\_def\\_t](#) \*csr\_def, uint8\_t \*csr, size\_t \*csr\_size)  
*Creates a CSR specified by the CSR definition from the ATECC508A device. This process involves reading the  
dynamic CSR data from the device and combining it with the template found in the CSR definition, then signing it.  
Return the CSR in der format.*

- int [atcacert\\_create\\_csr\\_pem](#) (const [atcacert\\_def\\_t](#) \*csr\_def, char \*csr, size\_t \*csr\_size)  
*Creates a CSR specified by the CSR definition from the ATECC508A device. This process involves reading the dynamic CSR data from the device and combining it with the template found in the CSR definition, then signing it. Return the CSR in der format.*
- int [atcacert\\_get\\_response](#) (uint8\_t device\_private\_key\_slot, const uint8\_t challenge[32], uint8\_t \*response[64])  
*Calculates the response to a challenge sent from the host.*
- int [atcacert\\_read\\_subj\\_key\\_id](#) (const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t subj\_key\_id[20])  
*Reads the subject key ID based on a certificate definition.*
- int [atcacert\\_read\\_cert\\_size](#) (const [atcacert\\_def\\_t](#) \*cert\_def, size\_t \*cert\_size)  
*Return the actual certificate size in bytes for a given cert def. Certificate can be variable size, so this gives the absolute buffer size when reading the certificates.*
- int [atcacert\\_date\\_enc](#) ([atcacert\\_date\\_format\\_t](#) format, const [atcacert\\_tm\\_utc\\_t](#) \*timestamp, uint8\_t \*formatted\_date, size\_t \*formatted\_date\_size)  
*Format a timestamp according to the format type.*
- int [atcacert\\_date\\_dec](#) ([atcacert\\_date\\_format\\_t](#) format, const uint8\_t \*formatted\_date, size\_t formatted\_date\_size, [atcacert\\_tm\\_utc\\_t](#) \*timestamp)  
*Parse a formatted timestamp according to the specified format.*
- int [atcacert\\_date\\_enc\\_compcert](#) (const [atcacert\\_tm\\_utc\\_t](#) \*issue\_date, uint8\_t expire\_years, uint8\_t enc\_dates[3])  
*Encode the issue and expire dates in the format used by the compressed certificate.*
- int [atcacert\\_date\\_dec\\_compcert](#) (const uint8\_t enc\_dates[3], [atcacert\\_date\\_format\\_t](#) expire\_date\_format, [atcacert\\_tm\\_utc\\_t](#) \*issue\_date, [atcacert\\_tm\\_utc\\_t](#) \*expire\_date)  
*Decode the issue and expire dates from the format used by the compressed certificate.*
- int [atcacert\\_date\\_get\\_max\\_date](#) ([atcacert\\_date\\_format\\_t](#) format, [atcacert\\_tm\\_utc\\_t](#) \*timestamp)  
*Return the maximum date available for the given format.*
- int [atcacert\\_date\\_enc\\_iso8601\\_sep](#) (const [atcacert\\_tm\\_utc\\_t](#) \*timestamp, uint8\_t formatted\_date[(20)])
- int [atcacert\\_date\\_dec\\_iso8601\\_sep](#) (const uint8\_t formatted\_date[(20)], [atcacert\\_tm\\_utc\\_t](#) \*timestamp)
- int [atcacert\\_date\\_enc\\_rfc5280\\_utc](#) (const [atcacert\\_tm\\_utc\\_t](#) \*timestamp, uint8\_t formatted\_date[(13)])
- int [atcacert\\_date\\_dec\\_rfc5280\\_utc](#) (const uint8\_t formatted\_date[(13)], [atcacert\\_tm\\_utc\\_t](#) \*timestamp)
- int [atcacert\\_date\\_enc\\_rfc5280\\_gen](#) (const [atcacert\\_tm\\_utc\\_t](#) \*timestamp, uint8\_t formatted\_date[(15)])
- int [atcacert\\_date\\_dec\\_rfc5280\\_gen](#) (const uint8\_t formatted\_date[(15)], [atcacert\\_tm\\_utc\\_t](#) \*timestamp)
- int [atcacert\\_date\\_enc\\_posix\\_uint32\\_be](#) (const [atcacert\\_tm\\_utc\\_t](#) \*timestamp, uint8\_t formatted\_date[(4)])
- int [atcacert\\_date\\_dec\\_posix\\_uint32\\_be](#) (const uint8\_t formatted\_date[(4)], [atcacert\\_tm\\_utc\\_t](#) \*timestamp)
- int [atcacert\\_date\\_enc\\_posix\\_uint32\\_le](#) (const [atcacert\\_tm\\_utc\\_t](#) \*timestamp, uint8\_t formatted\_date[(4)])
- int [atcacert\\_date\\_dec\\_posix\\_uint32\\_le](#) (const uint8\_t formatted\_date[(4)], [atcacert\\_tm\\_utc\\_t](#) \*timestamp)
- int [atcacert\\_get\\_device\\_locs](#) (const [atcacert\\_def\\_t](#) \*cert\_def, [atcacert\\_device\\_loc\\_t](#) \*device\_locs, size\_t \*device\_locs\_count, size\_t device\_locs\_max\_count, size\_t block\_size)  
*Add all the device locations required to rebuild the specified certificate (cert\_def) to a device locations list.*
- int [atcacert\\_cert\\_build\\_start](#) ([atcacert\\_build\\_state\\_t](#) \*build\_state, const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t \*cert\_size, const uint8\_t ca\_public\_key[64])  
*Starts the certificate rebuilding process.*
- int [atcacert\\_cert\\_build\\_process](#) ([atcacert\\_build\\_state\\_t](#) \*build\_state, const [atcacert\\_device\\_loc\\_t](#) \*device\_loc, const uint8\_t \*device\_data)  
*Process information read from the ATECC device. If it contains information for the certificate, it will be incorporated into the certificate.*
- int [atcacert\\_cert\\_build\\_finish](#) ([atcacert\\_build\\_state\\_t](#) \*build\_state)  
*Completes any final certificate processing required after all data from the device has been incorporated.*
- int [atcacert\\_get\\_device\\_data](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, const [atcacert\\_device\\_loc\\_t](#) \*device\_loc, uint8\_t \*device\_data)  
*Gets the dynamic data that would be saved to the specified device location. This function is primarily used to break down a full certificate into the dynamic components to be saved to a device.*
- int [atcacert\\_set\\_subj\\_public\\_key](#) (const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t cert\_size, const uint8\_t subj\_public\_key[64])

*Sets the subject public key and subject key ID in a certificate.*

- int `atcacert_get_subj_public_key` (const `atcacert_def_t` \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t subj\_public\_key[64])

*Gets the subject public key from a certificate.*

- int `atcacert_get_subj_key_id` (const `atcacert_def_t` \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t subj\_key\_id[20])

*Gets the subject key ID from a certificate.*

- int `atcacert_set_signature` (const `atcacert_def_t` \*cert\_def, uint8\_t \*cert, size\_t \*cert\_size, size\_t max\_cert\_size, const uint8\_t signature[64])

*Sets the signature in a certificate. This may alter the size of the X.509 certificates.*

- int `atcacert_get_signature` (const `atcacert_def_t` \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t signature[64])

*Gets the signature from a certificate.*

- int `atcacert_set_issue_date` (const `atcacert_def_t` \*cert\_def, uint8\_t \*cert, size\_t cert\_size, const `atcacert_tm_utc_t` \*timestamp)

*Sets the issue date (notBefore) in a certificate. Will be formatted according to the date format specified in the certificate definition.*

- int `atcacert_get_issue_date` (const `atcacert_def_t` \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, `atcacert_tm_utc_t` \*timestamp)

*Gets the issue date from a certificate. Will be parsed according to the date format specified in the certificate definition.*

- int `atcacert_set_expire_date` (const `atcacert_def_t` \*cert\_def, uint8\_t \*cert, size\_t cert\_size, const `atcacert_tm_utc_t` \*timestamp)

*Sets the expire date (notAfter) in a certificate. Will be formatted according to the date format specified in the certificate definition.*

- int `atcacert_get_expire_date` (const `atcacert_def_t` \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, `atcacert_tm_utc_t` \*timestamp)

*Gets the expire date from a certificate. Will be parsed according to the date format specified in the certificate definition.*

- int `atcacert_set_signer_id` (const `atcacert_def_t` \*cert\_def, uint8\_t \*cert, size\_t cert\_size, const uint8\_t signer\_id[2])

*Sets the signer ID in a certificate. Will be formatted as 4 upper-case hex digits.*

- int `atcacert_get_signer_id` (const `atcacert_def_t` \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t signer\_id[2])

*Gets the signer ID from a certificate. Will be parsed as 4 upper-case hex digits.*

- int `atcacert_set_cert_sn` (const `atcacert_def_t` \*cert\_def, uint8\_t \*cert, size\_t \*cert\_size, size\_t max\_cert\_size, const uint8\_t \*cert\_sn, size\_t cert\_sn\_size)

*Sets the certificate serial number in a certificate.*

- int `atcacert_gen_cert_sn` (const `atcacert_def_t` \*cert\_def, uint8\_t \*cert, size\_t cert\_size, const uint8\_t device\_sn[9])

*Sets the certificate serial number by generating it from other information in the certificate using the scheme specified by sn\_source in cert\_def. See the.*

- int `atcacert_get_cert_sn` (const `atcacert_def_t` \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t \*cert\_sn, size\_t cert\_sn\_size)

*Gets the certificate serial number from a certificate.*

- int `atcacert_set_auth_key_id` (const `atcacert_def_t` \*cert\_def, uint8\_t \*cert, size\_t cert\_size, const uint8\_t auth\_public\_key[64])

*Sets the authority key ID in a certificate. Note that this takes the actual public key creates a key ID from it.*

- int `atcacert_set_auth_key_id_raw` (const `atcacert_def_t` \*cert\_def, uint8\_t \*cert, size\_t cert\_size, const uint8\_t \*auth\_key\_id)

*Sets the authority key ID in a certificate.*

- int `atcacert_get_auth_key_id` (const `atcacert_def_t` \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t auth\_key\_id[20])

*Gets the authority key ID from a certificate.*

- int `atcacert_set_comp_cert` (const `atcacert_def_t` \*cert\_def, uint8\_t \*cert, size\_t \*cert\_size, size\_t max\_cert\_size, const uint8\_t comp\_cert[72])

*Sets the signature, issue date, expire date, and signer ID found in the compressed certificate. This also checks fields common between the cert\_def and the compressed certificate to make sure they match.*

- int **atcacert\_get\_comp\_cert** (const **atcacert\_def\_t** \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t ← t comp\_cert[72])

*Generate the compressed certificate for the given certificate.*

- int **atcacert\_get\_tbs** (const **atcacert\_def\_t** \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, const uint8\_t \*\*tbs, size\_t ← \*tbs\_size)

*Get a pointer to the TBS data in a certificate.*

- int **atcacert\_get\_tbs\_digest** (const **atcacert\_def\_t** \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t ← t tbs\_digest[32])

*Get the SHA256 digest of certificate's TBS data.*

- int **atcacert\_set\_cert\_element** (const **atcacert\_def\_t** \*cert\_def, const **atcacert\_cert\_loc\_t** \*cert\_loc, uint8\_t ← \*cert, size\_t cert\_size, const uint8\_t \*data, size\_t data\_size)

*Sets an element in a certificate. The data\_size must match the size in cert\_loc.*

- int **atcacert\_get\_cert\_element** (const **atcacert\_def\_t** \*cert\_def, const **atcacert\_cert\_loc\_t** \*cert\_loc, const uint8\_t ← \*cert, size\_t cert\_size, uint8\_t \*data, size\_t data\_size)

*Gets an element from a certificate.*

- int **atcacert\_get\_key\_id** (const uint8\_t public\_key[64], uint8\_t key\_id[20])

*Calculates the key ID for a given public ECC P256 key.*

- int **atcacert\_merge\_device\_loc** (**atcacert\_device\_loc\_t** \*device\_locs, size\_t \*device\_locs\_count, size\_t ← device\_locs\_max\_count, const **atcacert\_device\_loc\_t** \*device\_loc, size\_t block\_size)

*Merge a new device location into a list of device locations. If the new location overlaps with an existing location, the existing one will be modified to encompass both. Otherwise the new location is appended to the end of the list.*

- int **atcacert\_is\_device\_loc\_overlap** (const **atcacert\_device\_loc\_t** \*device\_loc1, const **atcacert\_device\_loc\_t** ← \*device\_loc2)

*Determines if the two device locations overlap.*

- void **atcacert\_public\_key\_add\_padding** (const uint8\_t raw\_key[64], uint8\_t padded\_key[72])

*Takes a raw P256 ECC public key and converts it to the padded version used by ATECC devices. Input and output buffers can point to the same location to do an in-place transform.*

- void **atcacert\_public\_key\_remove\_padding** (const uint8\_t padded\_key[72], uint8\_t raw\_key[64])

*Takes a padded public key used by ATECC devices and converts it to a raw P256 ECC public key. Input and output buffers can point to the same location to do an in-place transform.*

- int **atcacert\_transform\_data** (**atcacert\_transform\_t** transform, const uint8\_t \*data, size\_t data\_size, uint8\_t ← \*destination, size\_t \*destination\_size)

*Apply the specified transform to the specified data.*

- int **atcacert\_max\_cert\_size** (const **atcacert\_def\_t** \*cert\_def, size\_t \*max\_cert\_size)

*Return the maximum possible certificate size in bytes for a given cert def. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificates.*

- int **atcacert\_der\_enc\_length** (uint32\_t length, uint8\_t \*der\_length, size\_t \*der\_length\_size)

*Encode a length in DER format.*

- int **atcacert\_der\_dec\_length** (const uint8\_t \*der\_length, size\_t \*der\_length\_size, uint32\_t \*length)

*Decode a DER format length.*

- int **atcacert\_der\_adjust\_length** (uint8\_t \*der\_length, size\_t \*der\_length\_size, int delta\_length, uint32\_t ← \*new\_length)

- int **atcacert\_der\_enc\_integer** (const uint8\_t \*int\_data, size\_t int\_data\_size, uint8\_t is\_unsigned, uint8\_t ← \*der\_int, size\_t \*der\_int\_size)

*Encode an ASN.1 integer in DER format, including tag and length fields.*

- int **atcacert\_der\_dec\_integer** (const uint8\_t \*der\_int, size\_t \*der\_int\_size, uint8\_t \*int\_data, size\_t \*int ← data\_size)

*Decode an ASN.1 DER encoded integer.*

- int **atcacert\_der\_enc\_ecdsa\_sig\_value** (const uint8\_t raw\_sig[64], uint8\_t \*der\_sig, size\_t \*der\_sig\_size)

*Formats a raw ECDSA P256 signature in the DER encoding found in X.509 certificates.*

- int **atcacert\_der\_dec\_ecdsa\_sig\_value** (const uint8\_t \*der\_sig, size\_t \*der\_sig\_size, uint8\_t raw\_sig[64])



*Parses an ECDSA P256 signature in the DER encoding as found in X.509 certificates.*

- int [atcacert\\_verify\\_cert\\_hw](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, const uint8\_t ca\_public\_key[64])

*Verify a certificate against its certificate authority's public key using the host's ATECC device for crypto functions.*

- int [atcacert\\_gen\\_challenge\\_hw](#) (uint8\_t challenge[32])

*Generate a random challenge to be sent to the client using the RNG on the host's ATECC device.*

- int [atcacert\\_verify\\_response\\_hw](#) (const uint8\_t device\_public\_key[64], const uint8\_t challenge[32], const uint8\_t response[64])

*Verify a client's response to a challenge using the host's ATECC device for crypto functions.*

- int [atcacert\\_verify\\_cert\\_sw](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, const uint8\_t ca\_public\_key[64])

*Verify a certificate against its certificate authority's public key using software crypto functions. The function is currently not implemented.*

- int [atcacert\\_gen\\_challenge\\_sw](#) (uint8\_t challenge[32])

*Generate a random challenge to be sent to the client using a software PRNG. The function is currently not implemented.*

- int [atcacert\\_verify\\_response\\_sw](#) (const uint8\_t device\_public\_key[64], const uint8\_t challenge[32], const uint8\_t response[64])

*Verify a client's response to a challenge using software crypto functions. The function is currently not implemented.*

## Variables

- const size\_t [ATCACERT\\_DATE\\_FORMAT\\_SIZES](#) [5]

### 18.6.1 Detailed Description

These methods provide convenient ways to perform certification I/O with CryptoAuth chips and perform certificate manipulation in memory.

### 18.6.2 Macro Definition Documentation

#### 18.6.2.1 ATCA\_PACKED

```
#define ATCA_PACKED
```

#### 18.6.2.2 ATCACERT\_DATE\_FORMAT\_SIZES\_COUNT

```
#define ATCACERT_DATE_FORMAT_SIZES_COUNT 5
```

### 18.6.2.3 ATCACERT\_E\_BAD\_CERT

```
#define ATCACERT_E_BAD_CERT 10
```

Certificate structure is bad in some way.

### 18.6.2.4 ATCACERT\_E\_BAD\_PARAMS

```
#define ATCACERT_E_BAD_PARAMS 2
```

Invalid/bad parameter passed to function.

### 18.6.2.5 ATCACERT\_E\_BUFFER\_TOO\_SMALL

```
#define ATCACERT_E_BUFFER_TOO_SMALL 3
```

Supplied buffer for output is too small to hold the result.

### 18.6.2.6 ATCACERT\_E\_DECODING\_ERROR

```
#define ATCACERT_E_DECODING_ERROR 4
```

Data being decoded/parsed has an invalid format.

### 18.6.2.7 ATCACERT\_E\_ELEM\_MISSING

```
#define ATCACERT_E_ELEM_MISSING 8
```

The certificate element isn't defined for the certificate definition.

### 18.6.2.8 ATCACERT\_E\_ELEM\_OUT\_OF\_BOUNDS

```
#define ATCACERT_E_ELEM_OUT_OF_BOUNDS 9
```

Certificate element is out of bounds for the given certificate.

#### 18.6.2.9 ATCACERT\_E\_ERROR

```
#define ATCACERT_E_ERROR 1
```

General error.

#### 18.6.2.10 ATCACERT\_E\_INVALID\_DATE

```
#define ATCACERT_E_INVALID_DATE 5
```

Date is invalid.

#### 18.6.2.11 ATCACERT\_E\_INVALID\_TRANSFORM

```
#define ATCACERT_E_INVALID_TRANSFORM 13
```

Invalid transform passed to function.

#### 18.6.2.12 ATCACERT\_E\_SUCCESS

```
#define ATCACERT_E_SUCCESS 0
```

Operation completed successfully.

#### 18.6.2.13 ATCACERT\_E\_UNEXPECTED\_ELEM\_SIZE

```
#define ATCACERT_E_UNEXPECTED_ELEM_SIZE 7
```

A certificate element size was not what was expected.

#### 18.6.2.14 ATCACERT\_E\_UNIMPLEMENTED

```
#define ATCACERT_E_UNIMPLEMENTED 6
```

Function is unimplemented for the current configuration.

### 18.6.2.15 ATCACERT\_E\_VERIFY\_FAILED

```
#define ATCACERT_E_VERIFY_FAILED 12
```

Certificate or challenge/response verification failed.

### 18.6.2.16 ATCACERT\_E\_WRONG\_CERT\_DEF

```
#define ATCACERT_E_WRONG_CERT_DEF 11
```

### 18.6.2.17 DATEFMT\_ISO8601\_SEP\_SIZE

```
#define DATEFMT_ISO8601_SEP_SIZE (20)
```

### 18.6.2.18 DATEFMT\_MAX\_SIZE

```
#define DATEFMT_MAX_SIZE DATEFMT_ISO8601_SEP_SIZE
```

### 18.6.2.19 DATEFMT\_POSIX\_UINT32\_BE\_SIZE

```
#define DATEFMT_POSIX_UINT32_BE_SIZE (4)
```

### 18.6.2.20 DATEFMT\_POSIX\_UINT32\_LE\_SIZE

```
#define DATEFMT_POSIX_UINT32_LE_SIZE (4)
```

### 18.6.2.21 DATEFMT\_RFC5280\_GEN\_SIZE

```
#define DATEFMT_RFC5280_GEN_SIZE (15)
```

### 18.6.2.22 DATEFMT\_RFC5280\_UTC\_SIZE

```
#define DATEFMT_RFC5280_UTC_SIZE (13)
```

### 18.6.2.23 FALSE

```
#define FALSE (0)
```

### 18.6.2.24 TRUE

```
#define TRUE (1)
```

## 18.6.3 Typedef Documentation

### 18.6.3.1 atccert\_build\_state\_t

```
typedef struct atccert_build_state_s atccert_build_state_t
```

Tracks the state of a certificate as it's being rebuilt from device information.

### 18.6.3.2 atccert\_cert\_element\_t

```
typedef struct atccert_cert_element_s atccert_cert_element_t
```

Defines a generic dynamic element for a certificate including the device and template locations.

### 18.6.3.3 atccert\_cert\_loc\_t

```
typedef struct atccert_cert_loc_s atccert_cert_loc_t
```

Defines a chunk of data in a certificate template.

### 18.6.3.4 atccert\_cert\_sn\_src\_t

```
typedef enum atccert_cert_sn_src_e atccert_cert_sn_src_t
```

Sources for the certificate serial number.

### 18.6.3.5 atcacert\_cert\_type\_t

```
typedef enum atcacert_cert_type_e atcacert_cert_type_t
```

Types of certificates.

### 18.6.3.6 atcacert\_date\_format\_t

```
typedef enum atcacert_date_format_e atcacert_date_format_t
```

Date formats.

### 18.6.3.7 atcacert\_def\_t

```
typedef struct atcacert_def_s atcacert_def_t
```

Defines a certificate and all the pieces to work with it.

If any of the standard certificate elements (std\_cert\_elements) are not a part of the certificate definition, set their count to 0 to indicate their absence.

### 18.6.3.8 atcacert\_device\_loc\_t

```
typedef struct atcacert_device_loc_s atcacert_device_loc_t
```

Defines a chunk of data in an ATECC device.

### 18.6.3.9 atcacert\_device\_zone\_t

```
typedef enum atcacert_device_zone_e atcacert_device_zone_t
```

ATECC device zones. The values match the Zone Encodings as specified in the datasheet.

### 18.6.3.10 atcacert\_std\_cert\_element\_t

```
typedef enum atcacert_std_cert_element_e atcacert_std_cert_element_t
```

Standard dynamic certificate elements.

### 18.6.3.11 atcacert\_tm\_utc\_t

```
typedef struct atcacert_tm_utc_s atcacert_tm_utc_t
```

Holds a broken-down date in UTC. Mimics atcacert\_tm\_utc\_t from time.h.

### 18.6.3.12 atcacert\_transform\_t

```
typedef enum atcacert_transform_e atcacert_transform_t
```

How to transform the data from the device to the certificate.

## 18.6.4 Enumeration Type Documentation

### 18.6.4.1 atcacert\_cert\_sn\_src\_e

```
enum atcacert_cert_sn_src_e
```

Sources for the certificate serial number.

## 18.6 Certificate manipulation methods (atcacert\_)

### Enumerator

SNSRC_STORED	Cert serial is stored on the device.
SNSRC_STORED_DYNAMIC	Cert serial is stored on the device with the first byte being the DER size (X509 certs only).
SNSRC_DEVICE_SN	Cert serial number is 0x40(MSB) + 9-byte device serial number. Only applies to device certificates.
SNSRC_SIGNER_ID	Cert serial number is 0x40(MSB) + 2-byte signer ID. Only applies to signer certificates.
SNSRC_PUB_KEY_HASH	Cert serial number is the SHA256(Subject public key + Encoded dates), with uppermost 2 bits set to 01.
SNSRC_DEVICE_SN_HASH	Cert serial number is the SHA256(Device SN + Encoded dates), with uppermost 2 bits set to 01. Only applies to device certificates.
SNSRC_PUB_KEY_HASH_POS	Deprecated, don't use. Cert serial number is the SHA256(Subject public key + Encoded dates), with MSBit set to 0 to ensure it's positive.
SNSRC_DEVICE_SN_HASH_POS	Deprecated, don't use. Cert serial number is the SHA256(Device SN + Encoded dates), with MSBit set to 0 to ensure it's positive. Only applies to device certificates.
SNSRC_PUB_KEY_HASH_RAW	Deprecated, don't use. Cert serial number is the SHA256(Subject public key + Encoded dates).
SNSRC_DEVICE_SN_HASH_RAW	Deprecated, don't use. Cert serial number is the SHA256(Device SN + Encoded dates). Only applies to device certificates.

### 18.6.4.2 atcacert\_cert\_type\_e

enum `atcacert_cert_type_e`

Types of certificates.

### Enumerator

CERTTYPE_X509	Standard X509 certificate.
CERTTYPE_CUSTOM	Custom format.

### 18.6.4.3 atcacert\_date\_format\_e

enum `atcacert_date_format_e`

Date formats.

### Enumerator

DATEFMT_ISO8601_SEP	ISO8601 full date YYYY-MM-DDThh:mm:ssZ.
DATEFMT_RFC5280_UTC	RFC 5280 (X.509) 4.1.2.5.1 UTCTime format YYMMDDhhmmssZ.
DATEFMT_POSIX_UINT32_BE	POSIX (aka UNIX) date format. Seconds since Jan 1, 1970. 32 bit unsigned integer, big endian.



## Enumerator

DATEFMT_POSIX_UINT32_LE	POSIX (aka UNIX) date format. Seconds since Jan 1, 1970. 32 bit unsigned integer, little endian.
DATEFMT_RFC5280_GEN	RFC 5280 (X.509) 4.1.2.5.2 GeneralizedTime format YYYYMMDDhhmmssZ.

**18.6.4.4 atcacert\_device\_zone\_e**

enum `atcacert_device_zone_e`

ATECC device zones. The values match the Zone Encodings as specified in the datasheet.

## Enumerator

DEVZONE_CONFIG	Configuration zone.
DEVZONE_OTP	One Time Programmable zone.
DEVZONE_DATA	Data zone (slots).
DEVZONE_NONE	Special value used to indicate there is no device location.

**18.6.4.5 atcacert\_std\_cert\_element\_e**

enum `atcacert_std_cert_element_e`

Standard dynamic certificate elements.

## Enumerator

STDCERT_PUBLIC_KEY	
STDCERT_SIGNATURE	
STDCERT_ISSUE_DATE	
STDCERT_EXPIRE_DATE	
STDCERT_SIGNER_ID	
STDCERT_CERT_SN	
STDCERT_AUTH_KEY_ID	
STDCERT_SUBJ_KEY_ID	
STDCERT_NUM_ELEMENTS	Special item to give the number of elements in this enum.

**18.6.4.6 atcacert\_transform\_e**

enum `atcacert_transform_e`

## 18.6 Certificate manipulation methods (atcacert\_)

---

How to transform the data from the device to the certificate.

## Enumerator

TF_NONE	No transform, data is used byte for byte.
TF_REVERSE	Reverse the bytes (e.g. change endianness)
TF_BIN2HEX_UC	Convert raw binary into ASCII hex, uppercase.
TF_BIN2HEX_LC	Convert raw binary into ASCII hex, lowercase.
TF_HEX2BIN_UC	Convert ASCII hex, uppercase to binary.
TF_HEX2BIN_LC	Convert ASCII hex, lowercase to binary.
TF_BIN2HEX_SPACE_UC	Convert raw binary into ASCII hex, uppercase space between bytes.
TF_BIN2HEX_SPACE_LC	Convert raw binary into ASCII hex, lowercase space between bytes.
TF_HEX2BIN_SPACE_UC	Convert ASCII hex, uppercase with spaces between bytes to binary.
TF_HEX2BIN_SPACE_LC	Convert ASCII hex, lowercase with spaces between bytes to binary.

## 18.6.5 Function Documentation

### 18.6.5.1 atcacert\_cert\_build\_finish()

```
int atcacert_cert_build_finish (
    atcacert_build_state_t * build_state )
```

Completes any final certificate processing required after all data from the device has been incorporated.

The final certificate and its size in bytes are contained in the cert and cert\_size elements of the build\_state structure. This will be the same buffers as supplied to the atcacert\_cert\_build\_start function at the beginning of the certificate rebuilding process.

## Parameters

in	<i>build_state</i>	Current certificate build state.
----	--------------------	----------------------------------

## Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.2 atcacert\_cert\_build\_process()

```
int atcacert_cert_build_process (
    atcacert_build_state_t * build_state,
    const atcacert_device_loc_t * device_loc,
    const uint8_t * device_data )
```

Process information read from the ATECC device. If it contains information for the certificate, it will be incorporated into the certificate.

## 18.6 Certificate manipulation methods (atcacert\_)

### Parameters

in	<i>build_state</i>	Current certificate building state.
in	<i>device_loc</i>	Device location structure describing where on the device the following data came from.
in	<i>device_data</i>	Actual data from the device. It should represent the offset and byte count specified in the <i>device_loc</i> parameter.

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.3 atcacert\_cert\_build\_start()

```
int atcacert_cert_build_start (
    atcacert_build_state_t * build_state,
    const atcacert_def_t * cert_def,
    uint8_t * cert,
    size_t * cert_size,
    const uint8_t ca_public_key[64] )
```

Starts the certificate rebuilding process.

### Parameters

out	<i>build_state</i>	Structure is initialized to start the certificate building process. Will be passed to the other certificate building functions.
in	<i>cert_def</i>	Certificate definition for the certificate being built.
in	<i>cert</i>	Buffer to contain the rebuilt certificate.
in	<i>cert_size</i>	As input, the size of the cert buffer in bytes. This value will be adjusted to the current/final size of the certificate through the building process.
in	<i>ca_public_key</i>	ECC P256 public key of the certificate authority (issuer) for the certificate being built. Set to NULL if the authority key id is not needed, set properly in the <i>cert_def</i> template, or stored on the device as specified in the <i>cert_def</i> <i>cert_elements</i> .

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.4 atcacert\_create\_csr()

```
int atcacert_create_csr (
    const atcacert_def_t * csr_def,
    uint8_t * csr,
    size_t * csr_size )
```

Creates a CSR specified by the CSR definition from the ATECC508A device. This process involves reading the dynamic CSR data from the device and combining it with the template found in the CSR definition, then signing it. Return the CSR in der format.

**Parameters**

in	<i>csr_def</i>	CSR definition describing where to find the dynamic CSR information on the device and how to incorporate it into the template.
out	<i>csr</i>	Buffer to receive the CSR.
in, out	<i>csr_size</i>	As input, the size of the CSR buffer in bytes. As output, the size of the CSR returned in cert in bytes.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.6.5.5 atcacert\_create\_csr\_pem()**

```
int atcacert_create_csr_pem (
    const atcacert_def_t * csr_def,
    char * csr,
    size_t * csr_size )
```

Creates a CSR specified by the CSR definition from the ATECC508A device. This process involves reading the dynamic CSR data from the device and combining it with the template found in the CSR definition, then signing it. Return the CSR in der format.

**Parameters**

in	<i>csr_def</i>	CSR definition describing where to find the dynamic CSR information on the device and how to incorporate it into the template.
out	<i>csr</i>	Buffer to received the CSR formatted as PEM.
in, out	<i>csr_size</i>	As input, the size of the CSR buffer in bytes. As output, the size of the CSR as PEM returned in cert in bytes.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.6.5.6 atcacert\_date\_dec()**

```
int atcacert_date_dec (
    atcacert_date_format_t format,
    const uint8_t * formatted_date,
    size_t formatted_date_size,
    atcacert_tm_utc_t * timestamp )
```

Parse a formatted timestamp according to the specified format.

## 18.6 Certificate manipulation methods (atcacert\_)

---

### Parameters

in	<i>format</i>	Format to parse the formatted date as.
in	<i>formatted_date</i>	Formatted date to be parsed.
in	<i>formatted_date_size</i>	Size of the formatted date in bytes.
out	<i>timestamp</i>	Parsed timestamp is returned here.

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.7 atcacert\_date\_dec\_compcert()

```
int atcacert_date_dec_compcert (
    const uint8_t enc_dates[3],
    atcacert_date_format_t expire_date_format,
    atcacert_tm_utc_t * issue_date,
    atcacert_tm_utc_t * expire_date )
```

Decode the issue and expire dates from the format used by the compressed certificate.

### Parameters

in	<i>enc_dates</i>	Encoded date from the compressed certificate. 3 bytes.
in	<i>expire_date_format</i>	Expire date format. Only used to determine max date when no expiration date is specified by the encoded date.
out	<i>issue_date</i>	Decoded issue date is returned here.
out	<i>expire_date</i>	Decoded expire date is returned here. If there is no expiration date, the expire date will be set to a maximum value for the given <i>expire_date_format</i> .

### Returns

0 on success

### 18.6.5.8 atcacert\_date\_dec\_iso8601\_sep()

```
int atcacert_date_dec_iso8601_sep (
    const uint8_t formatted_date[(20)],
    atcacert_tm_utc_t * timestamp )
```

**18.6.5.9 atcacert\_date\_dec\_posix\_uint32\_be()**

```
int atcacert_date_dec_posix_uint32_be (
    const uint8_t formatted_date[(4)],
    atcacert_tm_utc_t * timestamp )
```

**18.6.5.10 atcacert\_date\_dec\_posix\_uint32\_le()**

```
int atcacert_date_dec_posix_uint32_le (
    const uint8_t formatted_date[(4)],
    atcacert_tm_utc_t * timestamp )
```

**18.6.5.11 atcacert\_date\_dec\_rfc5280\_gen()**

```
int atcacert_date_dec_rfc5280_gen (
    const uint8_t formatted_date[(15)],
    atcacert_tm_utc_t * timestamp )
```

**18.6.5.12 atcacert\_date\_dec\_rfc5280\_utc()**

```
int atcacert_date_dec_rfc5280_utc (
    const uint8_t formatted_date[(13)],
    atcacert_tm_utc_t * timestamp )
```

**18.6.5.13 atcacert\_date\_enc()**

```
int atcacert_date_enc (
    atcacert_date_format_t format,
    const atcacert_tm_utc_t * timestamp,
    uint8_t * formatted_date,
    size_t * formatted_date_size )
```

Format a timestamp according to the format type.

**Parameters**

in	<i>format</i>	Format to use.
in	<i>timestamp</i>	Timestamp to format.
out	<i>formatted_date</i>	Formatted date will be returned in this buffer.
in, out	<i>formatted_date_size</i>	As input, the size of the formatted_date buffer. As output, the size of the returned formatted_date.

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

#### 18.6.5.14 atcacert\_date\_enc\_compcert()

```
int atcacert_date_enc_compcert (
    const atcacert_tm_utc_t * issue_date,
    uint8_t expire_years,
    uint8_t enc_dates[3] )
```

Encode the issue and expire dates in the format used by the compressed certificate.

### Parameters

in	<i>issue_date</i>	Issue date to encode. Note that minutes and seconds will be ignored.
in	<i>expire_years</i>	Expire date is expressed as a number of years past the issue date. 0 should be used if there is no expire date.
out	<i>enc_dates</i>	Encoded dates for use in the compressed certificate is returned here. 3 bytes.

### Returns

0 on success

#### 18.6.5.15 atcacert\_date\_enc\_iso8601\_sep()

```
int atcacert_date_enc_iso8601_sep (
    const atcacert_tm_utc_t * timestamp,
    uint8_t formatted_date[(20)] )
```

#### 18.6.5.16 atcacert\_date\_enc\_posix\_uint32\_be()

```
int atcacert_date_enc_posix_uint32_be (
    const atcacert_tm_utc_t * timestamp,
    uint8_t formatted_date[(4)] )
```

#### 18.6.5.17 atcacert\_date\_enc\_posix\_uint32\_le()

```
int atcacert_date_enc_posix_uint32_le (
    const atcacert_tm_utc_t * timestamp,
    uint8_t formatted_date[(4)] )
```



#### 18.6.5.18 atcacert\_date\_enc\_rfc5280\_gen()

```
int atcacert_date_enc_rfc5280_gen (
    const atcacert_tm_utc_t * timestamp,
    uint8_t formatted_date[ (15)] )
```

#### 18.6.5.19 atcacert\_date\_enc\_rfc5280\_utc()

```
int atcacert_date_enc_rfc5280_utc (
    const atcacert_tm_utc_t * timestamp,
    uint8_t formatted_date[ (13)] )
```

#### 18.6.5.20 atcacert\_date\_get\_max\_date()

```
int atcacert_date_get_max_date (
    atcacert_date_format_t format,
    atcacert_tm_utc_t * timestamp )
```

Return the maximum date available for the given format.

##### Parameters

in	<i>format</i>	Format to get the max date for.
out	<i>timestamp</i>	Max date is returned here.

##### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

#### 18.6.5.21 atcacert\_der\_adjust\_length()

```
int atcacert_der_adjust_length (
    uint8_t * der_length,
    size_t * der_length_size,
    int delta_length,
    uint32_t * new_length )
```

## 18.6.5.22 atcacert\_der\_dec\_ecdsa\_sig\_value()

```
int atcacert_der_dec_ecdsa_sig_value (
    const uint8_t * der_sig,
    size_t * der_sig_size,
    uint8_t raw_sig[64] )
```

Parses an ECDSA P256 signature in the DER encoding as found in X.509 certificates.

This will parse the DER encoding of the signatureValue field as found in an X.509 certificate (RFC 5280). x509\_sig should include the tag, length, and value. The value of the signatureValue is the DER encoding of the ECDSA-Sig-Value as specified by RFC 5480 and SEC1.

## Parameters

in	<i>der_sig</i>	X.509 format signature (TLV of signatureValue) to be parsed.
in, out	<i>der_sig_size</i>	As input, size of the der_sig buffer in bytes. As output, size of the DER x.509 signature parsed from the buffer.
out	<i>raw_sig</i>	Parsed P256 ECDSA signature will be returned in this buffer. Formatted as R and S integers concatenated together. 64 bytes.

## Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

## 18.6.5.23 atcacert\_der\_dec\_integer()

```
int atcacert_der_dec_integer (
    const uint8_t * der_int,
    size_t * der_int_size,
    uint8_t * int_data,
    size_t * int_data_size )
```

Decode an ASN.1 DER encoded integer.

X.680 ( <http://www.itu.int/rec/T-REC-X.680/en>) section 19.8, for tag value X.690 ( <http://www.itu.int/rec/T-REC-X.690/en>) section 8.3, for encoding

## Parameters

in	<i>der_int</i>	DER encoded ASN.1 integer, including the tag and length fields.
in, out	<i>der_int_size</i>	As input, the size of the der_int buffer in bytes. As output, the size of the DER integer decoded in bytes.
out	<i>int_data</i>	Decode integer is returned in this buffer in a signed big-endian format.
in, out	<i>int_data_size</i>	As input, the size of int_data in bytes. As output, the size of the decoded integer in bytes.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.6.5.24 atcacert\_der\_dec\_length()**

```
int atcacert_der_dec_length (
    const uint8_t * der_length,
    size_t * der_length_size,
    uint32_t * length )
```

Decode a DER format length.

X.690 ( <http://www.itu.int/rec/T-REC-X.690/en>) section 8.1.3, for encoding

**Parameters**

in	<i>der_length</i>	DER encoded length.
in, out	<i>der_length_size</i>	As input, the size of the der_length buffer in bytes. As output, the size of the DER encoded length that was decoded.
out	<i>length</i>	Decoded length is returned here.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.6.5.25 atcacert\_der\_enc\_ecdsa\_sig\_value()**

```
int atcacert_der_enc_ecdsa_sig_value (
    const uint8_t raw_sig[64],
    uint8_t * der_sig,
    size_t * der_sig_size )
```

Formats a raw ECDSA P256 signature in the DER encoding found in X.509 certificates.

This will return the DER encoding of the signatureValue field as found in an X.509 certificate (RFC 5280). This include the tag, length, and value. The value of the signatureValue is the DER encoding of the ECDSA-Sig-Value as specified by RFC 5480 and SECG SEC1.

**Parameters**

in	<i>raw_sig</i>	P256 ECDSA signature to be formatted. Input format is R and S integers concatenated together. 64 bytes.
out	<i>der_sig</i>	X.509 format signature (TLV of signatureValue) will be returned in this buffer.
in, out	<i>der_sig_size</i>	As input, the size of the x509_sig buffer in bytes. As output, the size of the returned X.509 signature in bytes.

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

#### 18.6.5.26 atcacert\_der\_enc\_integer()

```
int atcacert_der_enc_integer (
    const uint8_t * int_data,
    size_t int_data_size,
    uint8_t is_unsigned,
    uint8_t * der_int,
    size_t * der_int_size )
```

Encode an ASN.1 integer in DER format, including tag and length fields.

X.680 ( <http://www.itu.int/rec/T-REC-X.680/en>) section 19.8, for tag value X.690 ( <http://www.itu.int/rec/T-REC-X.690/en>) section 8.3, for encoding

### Parameters

in	<i>int_data</i>	Raw integer in big-endian format.
in	<i>int_data_size</i>	Size of the raw integer in bytes.
in	<i>is_unsigned</i>	Indicate whether the input integer should be treated as unsigned.
out	<i>der_int</i>	DER encoded integer is returned in this buffer.
in, out	<i>der_int_size</i>	As input, the size of the der_int buffer in bytes. As output, the size of the DER integer returned in bytes.

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

#### 18.6.5.27 atcacert\_der\_enc\_length()

```
int atcacert_der_enc_length (
    uint32_t length,
    uint8_t * der_length,
    size_t * der_length_size )
```

Encode a length in DER format.

X.690 ( <http://www.itu.int/rec/T-REC-X.690/en>) section 8.1.3, for encoding

### Parameters

in	<i>length</i>	Length to be encoded.
out	<i>der_length</i>	DER encoded length will returned in this buffer.
in, out	<i>der_length_size</i>	As input, size of der_length buffer in bytes. As output, the size of the DER length encoding in bytes.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.6.5.28 atcacert\_gen\_cert\_sn()**

```
int atcacert_gen_cert_sn (
    const atcacert_def_t * cert_def,
    uint8_t * cert,
    size_t cert_size,
    const uint8_t device_sn[9] )
```

Sets the certificate serial number by generating it from other information in the certificate using the scheme specified by `sn_source` in `cert_def`. See the.

This method requires certain elements in the certificate be set properly as they're used for generating the serial number. See `atcacert_cert_sn_src_t` for what elements should be set in the certificate beforehand. If the `sn_source` is set to `SNSRC_STORED` or `SNSRC_STORED_DYNAMIC`, the function will return `ATCACERT_E_SUCCESS` without making any changes to the certificate.

**Parameters**

in	<i>cert_def</i>	Certificate definition for the certificate.
in, out	<i>cert</i>	Certificate to update.
in	<i>cert_size</i>	Size of the certificate ( <i>cert</i> ) in bytes.
in	<i>device_sn</i>	Device serial number, only used if required by the <code>sn_source</code> scheme. Can be set to NULL, if not required.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.6.5.29 atcacert\_gen\_challenge\_hw()**

```
int atcacert_gen_challenge_hw (
    uint8_t challenge[32] )
```

Generate a random challenge to be sent to the client using the RNG on the host's ATECC device.

**Parameters**

out	<i>challenge</i>	Random challenge is return here. 32 bytes.
-----	------------------	--

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.30 atcacert\_gen\_challenge\_sw()

```
int atcacert_gen_challenge_sw (
    uint8_t challenge[32] )
```

Generate a random challenge to be sent to the client using a software PRNG. The function is currently not implemented.

#### Parameters

out	<i>challenge</i>	Random challenge is return here. 32 bytes.
-----	------------------	--

#### Returns

ATCA\_UNIMPLEMENTED , as the function is currently not implemented.

### 18.6.5.31 atcacert\_get\_auth\_key\_id()

```
int atcacert_get_auth_key_id (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    uint8_t auth_key_id[20] )
```

Gets the authority key ID from a certificate.

#### Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>auth_key_id</i>	Authority key ID is returned in this buffer. 20 bytes.

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.32 atcacert\_get\_cert\_element()

```
int atcacert_get_cert_element (
    const atcacert_def_t * cert_def,
```

```

const atcacert_cert_loc_t * cert_loc,
const uint8_t * cert,
size_t cert_size,
uint8_t * data,
size_t data_size )

```

Gets an element from a certificate.

#### Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert_loc</i>	Certificate location for this element.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>data</i>	Element data will be returned in this buffer. This buffer must be large enough to hold cert_loc.count bytes.
in	<i>data_size</i>	Expected size of the cert element data.

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

#### 18.6.5.33 atcacert\_get\_cert\_sn()

```

int atcacert_get_cert_sn (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    uint8_t * cert_sn,
    size_t * cert_sn_size )

```

Gets the certificate serial number from a certificate.

#### Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>cert_sn</i>	Certificate SN will be returned in this buffer.
in, out	<i>cert_sn_size</i>	As input, the size of the cert_sn buffer. As output, the size of the certificate SN (cert_sn) in bytes.

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.34 atcacert\_get\_comp\_cert()

```
int atcacert_get_comp_cert (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    uint8_t comp_cert[72] )
```

Generate the compressed certificate for the given certificate.

#### Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to generate the compressed certificate for.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>comp_cert</i>	Compressed certificate is returned in this buffer. 72 bytes.

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.35 atcacert\_get\_device\_data()

```
int atcacert_get_device_data (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    const atcacert_device_loc_t * device_loc,
    uint8_t * device_data )
```

Gets the dynamic data that would be saved to the specified device location. This function is primarily used to break down a full certificate into the dynamic components to be saved to a device.

The `atcacert_add_device_locs` function can be used to generate a list of device locations a particular certificate definition requires.

#### Parameters

in	<i>cert_def</i>	Certificate definition for the certificate we're getting data from.
in	<i>cert</i>	Certificate to get the device data from.
in	<i>cert_size</i>	Size of the certificate in bytes.
in	<i>device_loc</i>	Device location to request data for.
out	<i>device_data</i>	Buffer that represents the device data in <code>device_loc</code> . Required to be at least <code>device_loc.count</code> in size.

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.



### 18.6.5.36 atcacert\_get\_device\_locs()

```
int atcacert_get_device_locs (
    const atcacert_def_t * cert_def,
    atcacert_device_loc_t * device_locs,
    size_t * device_locs_count,
    size_t device_locs_max_count,
    size_t block_size )
```

Add all the device locations required to rebuild the specified certificate (*cert\_def*) to a device locations list.

The *block\_size* parameter will adjust all added device locations to have a offset and count that aligns with that block size. This allows one to generate a list of device locations that matches specific read or write semantics (e.g. 4 byte or 32 byte reads).

#### Parameters

in	<i>cert_def</i>	Certificate definition containing all the device locations to add to the list.
in, out	<i>device_locs</i>	List of device locations to add to.
in, out	<i>device_locs_count</i>	As input, existing size of the device locations list. As output, the new size of the device locations list.
in	<i>device_locs_max_count</i>	Maximum number of elements <i>device_locs</i> can hold.
in	<i>block_size</i>	Block size to align all offsets and counts to when adding device locations.

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.37 atcacert\_get\_expire\_date()

```
int atcacert_get_expire_date (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    atcacert_tm_utc_t * timestamp )
```

Gets the expire date from a certificate. Will be parsed according to the date format specified in the certificate definition.

#### Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate ( <i>cert</i> ) in bytes.
out	<i>timestamp</i>	Expire date is returned in this structure.

## 18.6 Certificate manipulation methods (atcacert\_)

---

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

#### 18.6.5.38 atcacert\_get\_issue\_date()

```
int atcacert_get_issue_date (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    atcacert_tm_utc_t * timestamp )
```

Gets the issue date from a certificate. Will be parsed according to the date format specified in the certificate definition.

### Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>timestamp</i>	Issue date is returned in this structure.

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

#### 18.6.5.39 atcacert\_get\_key\_id()

```
int atcacert_get_key_id (
    const uint8_t public_key[64],
    uint8_t key_id[20] )
```

Calculates the key ID for a given public ECC P256 key.

Uses method 1 for calculating the keyIdentifier as specified by RFC 5280, section 4.2.1.2: (1) The keyIdentifier is composed of the 160-bit SHA-1 hash of the value of the BIT STRING subjectPublicKey (excluding the tag, length, and number of unused bits).

### Parameters

in	<i>public_key</i>	ECC P256 public key to calculate key key ID for. Formatted as the X and Y integers concatenated together. 64 bytes.
in	<i>key_id</i>	Calculated key ID will be returned in this buffer. 20 bytes.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.6.5.40 atcacert\_get\_response()**

```
int atcacert_get_response (
    uint8_t device_private_key_slot,
    const uint8_t challenge[32],
    uint8_t response[64] )
```

Calculates the response to a challenge sent from the host.

The challenge-response protocol is an ECDSA Sign and Verify. This performs the ECDSA Sign on the challenge and returns the signature as the response.

**Parameters**

in	<i>device_private_key_slot</i>	Slot number for the device's private key. This must be the same slot used to generate the public key included in the device's certificate.
in	<i>challenge</i>	Challenge to generate the response for. Must be 32 bytes.
out	<i>response</i>	Response will be returned in this buffer. 64 bytes.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.6.5.41 atcacert\_get\_signature()**

```
int atcacert_get_signature (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    uint8_t signature[64] )
```

Gets the signature from a certificate.

**Parameters**

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>signature</i>	Signature is returned in this buffer. Formatted at R and S integers concatenated together. 64 bytes.

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

#### 18.6.5.42 atcacert\_get\_signer\_id()

```
int atcacert_get_signer_id (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    uint8_t signer_id[2] )
```

Gets the signer ID from a certificate. Will be parsed as 4 upper-case hex digits.

### Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>signer_id</i>	Signer ID will be returned in this buffer. 2 bytes.

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

#### 18.6.5.43 atcacert\_get\_subj\_key\_id()

```
int atcacert_get_subj_key_id (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    uint8_t subj_key_id[20] )
```

Gets the subject key ID from a certificate.

### Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>subj_key_id</i>	Subject key ID is returned in this buffer. 20 bytes.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.6.5.44 atcacert\_get\_subj\_public\_key()**

```
int atcacert_get_subj_public_key (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    uint8_t subj_public_key[64] )
```

Gets the subject public key from a certificate.

**Parameters**

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get element from.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>subj_public_key</i>	Subject public key is returned in this buffer. Formatted at X and Y integers concatenated together. 64 bytes.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.6.5.45 atcacert\_get\_tbs()**

```
int atcacert_get_tbs (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    const uint8_t ** tbs,
    size_t * tbs_size )
```

Get a pointer to the TBS data in a certificate.

**Parameters**

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get the TBS data pointer for.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>tbs</i>	Pointer to a const pointer that will be set the start of the TBS data.
out	<i>tbs_size</i>	Size of the TBS data will be returned here.

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

#### 18.6.5.46 atcacert\_get\_tbs\_digest()

```
int atcacert_get_tbs_digest (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    uint8_t tbs_digest[32] )
```

Get the SHA256 digest of certificate's TBS data.

### Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert</i>	Certificate to get the TBS data pointer for.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
out	<i>tbs_digest</i>	TBS data digest will be returned here. 32 bytes.

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

#### 18.6.5.47 atcacert\_is\_device\_loc\_overlap()

```
int atcacert_is_device_loc_overlap (
    const atcacert_device_loc_t * device_loc1,
    const atcacert_device_loc_t * device_loc2 )
```

Determines if the two device locations overlap.

### Parameters

in	<i>device_loc1</i>	First device location to check.
in	<i>device_loc2</i>	Second device location o check.

### Returns

0 (false) if they don't overlap, non-zero if the do overlap.

**18.6.5.48 atcacert\_max\_cert\_size()**

```
int atcacert_max_cert_size (
    const atcacert_def_t * cert_def,
    size_t * max_cert_size )
```

Return the maximum possible certificate size in bytes for a given cert def. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificates.

**Parameters**

in	<i>cert_def</i>	Certificate definition to find a max size for.
out	<i>max_cert_size</i>	Maximum certificate size will be returned here in bytes.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.6.5.49 atcacert\_merge\_device\_loc()**

```
int atcacert_merge_device_loc (
    atcacert_device_loc_t * device_locs,
    size_t * device_locs_count,
    size_t device_locs_max_count,
    const atcacert_device_loc_t * device_loc,
    size_t block_size )
```

Merge a new device location into a list of device locations. If the new location overlaps with an existing location, the existing one will be modified to encompass both. Otherwise the new location is appended to the end of the list.

The block\_size parameter will adjust all added device locations to have an offset and count that aligns with that block size. This allows one to generate a list of device locations that matches specific read/write semantics (e.g. 4 byte or 32 byte reads). Note that this block\_size only applies to the device\_loc being added. Existing device locations in the list won't be modified to match the block size.

**Parameters**

in, out	<i>device_locs</i>	Existing device location list to merge the new device location into.
in, out	<i>device_locs_count</i>	As input, the existing number of items in the device_locs list. As output, the new size of the device_locs list.
in	<i>device_locs_max_count</i>	Maximum number of items the device_locs list can hold.
in	<i>device_loc</i>	New device location to be merged into the device_locs list.
in	<i>block_size</i>	Block size to align all offsets and counts to when adding device location.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

## 18.6 Certificate manipulation methods (atcacert\_)

---

### 18.6.5.50 atcacert\_public\_key\_add\_padding()

```
void atcacert_public_key_add_padding (
    const uint8_t raw_key[64],
    uint8_t padded_key[72] )
```

Takes a raw P256 ECC public key and converts it to the padded version used by ATECC devices. Input and output buffers can point to the same location to do an in-place transform.

#### Parameters

in	<i>raw_key</i>	Public key as X and Y integers concatenated together. 64 bytes.
out	<i>padded_key</i>	Padded key is returned in this buffer. X and Y integers are padded with 4 bytes of 0 in the MSB. 72 bytes.

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.51 atcacert\_public\_key\_remove\_padding()

```
void atcacert_public_key_remove_padding (
    const uint8_t padded_key[72],
    uint8_t raw_key[64] )
```

Takes a padded public key used by ATECC devices and converts it to a raw P256 ECC public key. Input and output buffers can point to the same location to do an in-place transform.

#### Parameters

out	<i>padded_key</i>	X and Y integers are padded with 4 bytes of 0 in the MSB. 72 bytes.
in	<i>raw_key</i>	Raw key is returned in this buffer. Public key as X and Y integers concatenated together. 64 bytes.

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.52 atcacert\_read\_cert()

```
int atcacert_read_cert (
    const atcacert_def_t * cert_def,
    const uint8_t ca_public_key[64],
    uint8_t * cert,
    size_t * cert_size )
```



Reads the certificate specified by the certificate definition from the ATECC508A device.

This process involves reading the dynamic cert data from the device and combining it with the template found in the certificate definition.

## 18.6 Certificate manipulation methods (atccert\_)

### Parameters

in	<i>cert_def</i>	Certificate definition describing where to find the dynamic certificate information on the device and how to incorporate it into the template.
in	<i>ca_public_key</i>	The ECC P256 public key of the certificate authority that signed this certificate. Formatted as the 32 byte X and Y integers concatenated together (64 bytes total). Set to NULL if the authority key id is not needed, set properly in the cert_def template, or stored on the device as specified in the cert_def cert_elements.
out	<i>cert</i>	Buffer to received the certificate.
in, out	<i>cert_size</i>	As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in cert in bytes.

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

#### 18.6.5.53 atccert\_read\_cert\_size()

```
int atccert_read_cert_size (
    const atccert_def_t * cert_def,
    size_t * cert_size )
```

Return the actual certificate size in bytes for a given cert def. Certificate can be variable size, so this gives the absolute buffer size when reading the certificates.

### Parameters

in	<i>cert_def</i>	Certificate definition to find a max size for.
out	<i>cert_size</i>	Certificate size will be returned here in bytes.

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

#### 18.6.5.54 atccert\_read\_device\_loc()

```
int atccert_read_device_loc (
    const atccert_device_loc_t * device_loc,
    uint8_t * data )
```

Read the data from a device location.

### Parameters

in	<i>device_loc</i>	Device location to read data from.
out	<i>data</i>	Data read is returned here.

Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

18.6.5.55 atcacert\_read\_subj\_key\_id()

```
int atcacert_read_subj_key_id (
    const atcacert_def_t * cert_def,
    uint8_t subj_key_id[20] )
```

Reads the subject key ID based on a certificate definition.

Parameters

in	<i>cert_def</i>	Certificate definition
out	<i>subj_key_id</i>	Subject key ID is returned in this buffer. 20 bytes.

Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

18.6.5.56 atcacert\_set\_auth\_key\_id()

```
int atcacert_set_auth_key_id (
    const atcacert_def_t * cert_def,
    uint8_t * cert,
    size_t cert_size,
    const uint8_t auth_public_key[64] )
```

Sets the authority key ID in a certificate. Note that this takes the actual public key creates a key ID from it.

Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in, out	<i>cert</i>	Certificate to update.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
in	<i>auth_public_key</i>	Authority public key as X and Y integers concatenated together. 64 bytes.

Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.57 atcacert\_set\_auth\_key\_id\_raw()

```
int atcacert_set_auth_key_id_raw (
    const atcacert_def_t * cert_def,
    uint8_t * cert,
    size_t cert_size,
    const uint8_t * auth_key_id )
```

Sets the authority key ID in a certificate.

#### Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in, out	<i>cert</i>	Certificate to update.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
in	<i>auth_key_id</i>	Authority key ID. Same size as defined in the cert_def.

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.58 atcacert\_set\_cert\_element()

```
int atcacert_set_cert_element (
    const atcacert_def_t * cert_def,
    const atcacert_cert_loc_t * cert_loc,
    uint8_t * cert,
    size_t cert_size,
    const uint8_t * data,
    size_t data_size )
```

Sets an element in a certificate. The data\_size must match the size in cert\_loc.

#### Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in	<i>cert_loc</i>	Certificate location for this element.
in, out	<i>cert</i>	Certificate to update.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
in	<i>data</i>	Element data to insert into the certificate. Buffer must contain cert_loc.count bytes to be copied into the certificate.
in	<i>data_size</i>	Size of the data in bytes.

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.6.5.59 atcacert\_set\_cert\_sn()**

```
int atcacert_set_cert_sn (
    const atcacert_def_t * cert_def,
    uint8_t * cert,
    size_t * cert_size,
    size_t max_cert_size,
    const uint8_t * cert_sn,
    size_t cert_sn_size )
```

Sets the certificate serial number in a certificate.

**Parameters**

in	<i>cert_def</i>	Certificate definition for the certificate.
in, out	<i>cert</i>	Certificate to update.
in, out	<i>cert_size</i>	Size of the certificate (cert) in bytes.
in	<i>max_cert_size</i>	Maximum size of the cert buffer.
in	<i>cert_sn</i>	Certificate serial number.
in	<i>cert_sn_size</i>	Size of the certificate serial number in bytes.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.6.5.60 atcacert\_set\_comp\_cert()**

```
int atcacert_set_comp_cert (
    const atcacert_def_t * cert_def,
    uint8_t * cert,
    size_t * cert_size,
    size_t max_cert_size,
    const uint8_t comp_cert[72] )
```

Sets the signature, issue date, expire date, and signer ID found in the compressed certificate. This also checks fields common between the cert\_def and the compressed certificate to make sure they match.

**Parameters**

in	<i>cert_def</i>	Certificate definition for the certificate.
in, out	<i>cert</i>	Certificate to update.
in, out	<i>cert_size</i>	As input, size of the certificate (cert) in bytes. As output, the new size of the certificate.
in	<i>max_cert_size</i>	Maximum size of the cert buffer.
in	<i>comp_cert</i>	Compressed certificate. 72 bytes.

## 18.6 Certificate manipulation methods (atcacert\_)

---

### Returns

ATCACERT\_E\_SUCCESS on success. ATCACERT\_E\_WRONG\_CERT\_DEF if the template ID, chain ID, and/or SN source don't match between the cert\_def and the compressed certificate.

### 18.6.5.61 atcacert\_set\_expire\_date()

```
int atcacert_set_expire_date (
    const atcacert_def_t * cert_def,
    uint8_t * cert,
    size_t cert_size,
    const atcacert_tm_utc_t * timestamp )
```

Sets the expire date (notAfter) in a certificate. Will be formatted according to the date format specified in the certificate definition.

### Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in, out	<i>cert</i>	Certificate to update.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
in	<i>timestamp</i>	Expire date.

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.62 atcacert\_set\_issue\_date()

```
int atcacert_set_issue_date (
    const atcacert_def_t * cert_def,
    uint8_t * cert,
    size_t cert_size,
    const atcacert_tm_utc_t * timestamp )
```

Sets the issue date (notBefore) in a certificate. Will be formatted according to the date format specified in the certificate definition.

### Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in, out	<i>cert</i>	Certificate to update.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
in	<i>timestamp</i>	Issue date.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.6.5.63 atcacert\_set\_signature()**

```
int atcacert_set_signature (
    const atcacert_def_t * cert_def,
    uint8_t * cert,
    size_t * cert_size,
    size_t max_cert_size,
    const uint8_t signature[64] )
```

Sets the signature in a certificate. This may alter the size of the X.509 certificates.

**Parameters**

in	<i>cert_def</i>	Certificate definition for the certificate.
in, out	<i>cert</i>	Certificate to update.
in, out	<i>cert_size</i>	As input, size of the certificate (cert) in bytes. As output, the new size of the certificate.
in	<i>max_cert_size</i>	Maximum size of the cert buffer.
in	<i>signature</i>	Signature as R and S integers concatenated together. 64 bytes.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.6.5.64 atcacert\_set\_signer\_id()**

```
int atcacert_set_signer_id (
    const atcacert_def_t * cert_def,
    uint8_t * cert,
    size_t cert_size,
    const uint8_t signer_id[2] )
```

Sets the signer ID in a certificate. Will be formatted as 4 upper-case hex digits.

**Parameters**

in	<i>cert_def</i>	Certificate definition for the certificate.
in, out	<i>cert</i>	Certificate to update.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
in	<i>signer_id</i>	Signer ID.

## 18.6 Certificate manipulation methods (atcacert\_)

---

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.65 atcacert\_set\_subj\_public\_key()

```
int atcacert_set_subj_public_key (
    const atcacert_def_t * cert_def,
    uint8_t * cert,
    size_t cert_size,
    const uint8_t subj_public_key[64] )
```

Sets the subject public key and subject key ID in a certificate.

### Parameters

in	<i>cert_def</i>	Certificate definition for the certificate.
in, out	<i>cert</i>	Certificate to update.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
in	<i>subj_public_key</i>	Subject public key as X and Y integers concatenated together. 64 bytes.

### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.6.5.66 atcacert\_transform\_data()

```
int atcacert_transform_data (
    atcacert_transform_t transform,
    const uint8_t * data,
    size_t data_size,
    uint8_t * destination,
    size_t * destination_size )
```

Apply the specified transform to the specified data.

### Parameters

in	<i>transform</i>	Transform to be performed.
in	<i>data</i>	Input data to be transformed.
in	<i>data_size</i>	Size of the input data in bytes.
out	<i>destination</i>	Destination buffer to hold the transformed data.
in, out	<i>destination_size</i>	As input, the size of the destination buffer. As output the size of the transformed data.



**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.6.5.67 atcacert\_verify\_cert\_hw()**

```
int atcacert_verify_cert_hw (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    const uint8_t ca_public_key[64] )
```

Verify a certificate against its certificate authority's public key using the host's ATECC device for crypto functions.

**Parameters**

in	<i>cert_def</i>	Certificate definition describing how to extract the TBS and signature components from the certificate specified.
in	<i>cert</i>	Certificate to verify.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
in	<i>ca_public_key</i>	The ECC P256 public key of the certificate authority that signed this certificate. Formatted as the 32 byte X and Y integers concatenated together (64 bytes total).

**Returns**

ATCACERT\_E\_SUCCESS if the verify succeeds, ATCACERT\_VERIFY\_FAILED or ATCA\_EXECUTION\_ERROR if it fails to verify. ATCA\_EXECUTION\_ERROR may occur when the public key is invalid and doesn't fall on the P256 curve.

**18.6.5.68 atcacert\_verify\_cert\_sw()**

```
int atcacert_verify_cert_sw (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size,
    const uint8_t ca_public_key[64] )
```

Verify a certificate against its certificate authority's public key using software crypto functions. The function is currently not implemented.

**Parameters**

in	<i>cert_def</i>	Certificate definition describing how to extract the TBS and signature components from the certificate specified.
in	<i>cert</i>	Certificate to verify.
in	<i>cert_size</i>	Size of the certificate (cert) in bytes.
in	<i>ca_public_key</i>	The ECC P256 public key of the certificate authority that signed this certificate. Formatted as the 32 byte X and Y integers concatenated together (64 bytes total).

## 18.6 Certificate manipulation methods (atcacert\_)

### Returns

ATCA\_UNIMPLEMENTED , as the function is currently not implemented.

### 18.6.5.69 atcacert\_verify\_response\_hw()

```
int atcacert_verify_response_hw (
    const uint8_t device_public_key[64],
    const uint8_t challenge[32],
    const uint8_t response[64] )
```

Verify a client's response to a challenge using the host's ATECC device for crypto functions.

The challenge-response protocol is an ECDSA Sign and Verify. This performs an ECDSA verify on the response returned by the client, verifying the client has the private key counter-part to the public key returned in its certificate.

### Parameters

in	<i>device_public_key</i>	Device public key as read from its certificate. Formatted as the X and Y integers concatenated together. 64 bytes.
in	<i>challenge</i>	Challenge that was sent to the client. 32 bytes.
in	<i>response</i>	Response returned from the client to be verified. 64 bytes.

### Returns

ATCACERT\_E\_SUCCESS if the verify succeeds, ATCACERT\_VERIFY\_FAILED or ATCA\_EXECUTION\_ERROR if it fails to verify. ATCA\_EXECUTION\_ERROR may occur when the public key is invalid and doesn't fall on the P256 curve.

### 18.6.5.70 atcacert\_verify\_response\_sw()

```
int atcacert_verify_response_sw (
    const uint8_t device_public_key[64],
    const uint8_t challenge[32],
    const uint8_t response[64] )
```

Verify a client's response to a challenge using software crypto functions. The function is currently not implemented.

The challenge-response protocol is an ECDSA Sign and Verify. This performs an ECDSA verify on the response returned by the client, verifying the client has the private key counter-part to the public key returned in its certificate.

### Parameters

in	<i>device_public_key</i>	Device public key as read from its certificate. Formatted as the X and Y integers concatenated together. 64 bytes.
in	<i>challenge</i>	Challenge that was sent to the client. 32 bytes.
in	<i>response</i>	Response returned from the client to be verified. 64 bytes.

**Returns**

ATCA\_UNIMPLEMENTED , as the function is currently not implemented.

**18.6.5.71 atcacert\_write\_cert()**

```
int atcacert_write_cert (
    const atcacert_def_t * cert_def,
    const uint8_t * cert,
    size_t cert_size )
```

Take a full certificate and write it to the ATECC508A device according to the certificate definition.

**Parameters**

in	<i>cert_def</i>	Certificate definition describing where the dynamic certificate information is and how to store it on the device.
in	<i>cert</i>	Full certificate to be stored.
in	<i>cert_size</i>	Size of the full certificate in bytes.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.6.6 Variable Documentation****18.6.6.1 ATCACERT\_DATE\_FORMAT\_SIZES**

```
const size_t ATCACERT_DATE_FORMAT_SIZES[5]
```

## 18.7 Basic Crypto API methods for CryptoAuth Devices (calib\_)

These methods provide a simple API to CryptoAuth chips.

### Data Structures

- struct [atca\\_sha256\\_ctx](#)

### Typedefs

- typedef struct [atca\\_sha256\\_ctx](#) [atca\\_sha256\\_ctx\\_t](#)
- typedef [atca\\_sha256\\_ctx\\_t](#) [atca\\_hmac\\_sha256\\_ctx\\_t](#)

### Functions

- [ATCA\\_STATUS calib\\_wakeup](#) ([ATCADevice](#) device)  
*basic API methods are all prefixed with atcab\_ (CryptoAuthLib Basic) the fundamental premise of the basic API is it is based on a single interface instance and that instance is global, so all basic API commands assume that one global device is the one to operate on.*
- [ATCA\\_STATUS calib\\_idle](#) ([ATCADevice](#) device)  
*idle the CryptoAuth device*
- [ATCA\\_STATUS calib\\_sleep](#) ([ATCADevice](#) device)  
*invoke sleep on the CryptoAuth device*
- [ATCA\\_STATUS \\_calib\\_exit](#) ([ATCADevice](#) device)  
*common cleanup code which idles the device after any operation*
- [ATCA\\_STATUS calib\\_cfg\\_discover](#) ([ATCAIfaceCfg](#) cfg\_array[], int max)  
*auto discovery of crypto auth devices*
- [ATCA\\_STATUS calib\\_get\\_addr](#) (uint8\_t zone, uint16\_t slot, uint8\_t block, uint8\_t offset, uint16\_t \*addr)  
*Compute the address given the zone, slot, block, and offset.*
- [ATCA\\_STATUS calib\\_get\\_zone\\_size](#) ([ATCADevice](#) device, uint8\_t zone, uint16\_t slot, size\_t \*size)  
*Gets the size of the specified zone in bytes.*
- [ATCA\\_STATUS calib\\_aes](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*aes\_in, uint8\_t \*aes\_out)  
*Compute the AES-128 encrypt, decrypt, or GFM calculation.*
- [ATCA\\_STATUS calib\\_aes\\_encrypt](#) ([ATCADevice](#) device, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*plaintext, uint8\_t \*ciphertext)  
*Perform an AES-128 encrypt operation with a key in the device.*
- [ATCA\\_STATUS calib\\_aes\\_decrypt](#) ([ATCADevice](#) device, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*ciphertext, uint8\_t \*plaintext)  
*Perform an AES-128 decrypt operation with a key in the device.*
- [ATCA\\_STATUS calib\\_aes\\_gfm](#) ([ATCADevice](#) device, const uint8\_t \*h, const uint8\_t \*input, uint8\_t \*output)  
*Perform a Galois Field Multiply (GFM) operation.*
- [ATCA\\_STATUS calib\\_checkmac](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*challenge, const uint8\_t \*response, const uint8\_t \*other\_data)  
*Compares a MAC response with input values.*
- [ATCA\\_STATUS calib\\_counter](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Compute the Counter functions.*
- [ATCA\\_STATUS calib\\_counter\\_increment](#) ([ATCADevice](#) device, uint16\_t counter\_id, uint32\_t \*counter\_value)

- Increments one of the device's monotonic counters.*

  - **ATCA\_STATUS calib\_counter\_read** (ATCADevice device, uint16\_t counter\_id, uint32\_t \*counter\_value)

*Read one of the device's monotonic counters.*
- **ATCA\_STATUS calib\_derivekey** (ATCADevice device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*mac)

*Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.*
- **ATCA\_STATUS calib\_ecdh\_base** (ATCADevice device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, uint8\_t \*out\_nonce)

*Base function for generating premaster secret key using ECDH.*
- **ATCA\_STATUS calib\_ecdh** (ATCADevice device, uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms)

*ECDH command with a private key in a slot and the premaster secret is returned in the clear.*
- **ATCA\_STATUS calib\_ecdh\_enc** (ATCADevice device, uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*read\_key, uint16\_t read\_key\_id, const uint8\_t num\_in[(20)])
- **ATCA\_STATUS calib\_ecdh\_ioenc** (ATCADevice device, uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*io\_key)

*ECDH command with a private key in a slot and the premaster secret is returned encrypted using the IO protection key.*
- **ATCA\_STATUS calib\_ecdh\_tempkey** (ATCADevice device, const uint8\_t \*public\_key, uint8\_t \*pms)

*ECDH command with a private key in TempKey and the premaster secret is returned in the clear.*
- **ATCA\_STATUS calib\_ecdh\_tempkey\_ioenc** (ATCADevice device, const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*io\_key)

*ECDH command with a private key in TempKey and the premaster secret is returned encrypted using the IO protection key.*
- **ATCA\_STATUS calib\_gendig** (ATCADevice device, uint8\_t zone, uint16\_t key\_id, const uint8\_t \*other\_data, uint8\_t other\_data\_size)

*Issues a GenDig command, which performs a SHA256 hash on the source data indicated by zone with the contents of TempKey. See the CryptoAuth datasheet for your chip to see what the values of zone correspond to.*
- **ATCA\_STATUS calib\_genkey\_base** (ATCADevice device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*other\_data, uint8\_t \*public\_key)

*Issues GenKey command, which can generate a private key, compute a public key, nd/or compute a digest of a public key.*
- **ATCA\_STATUS calib\_genkey** (ATCADevice device, uint16\_t key\_id, uint8\_t \*public\_key)

*Issues GenKey command, which generates a new random private key in slot and returns the public key.*
- **ATCA\_STATUS calib\_get\_pubkey** (ATCADevice device, uint16\_t key\_id, uint8\_t \*public\_key)

*Uses GenKey command to calculate the public key from an existing private key in a slot.*
- **ATCA\_STATUS calib\_hmac** (ATCADevice device, uint8\_t mode, uint16\_t key\_id, uint8\_t \*digest)

*Issues a HMAC command, which computes an HMAC/SHA-256 digest of a key stored in the device, a challenge, and other information on the device.*
- **ATCA\_STATUS calib\_info\_base** (ATCADevice device, uint8\_t mode, uint16\_t param2, uint8\_t \*out\_data)

*Issues an Info command, which return internal device information and can control GPIO and the persistent latch.*
- **ATCA\_STATUS calib\_info** (ATCADevice device, uint8\_t \*revision)

*Use the Info command to get the device revision (DevRev).*
- **ATCA\_STATUS calib\_info\_set\_latch** (ATCADevice device, bool state)

*Use the Info command to set the persistent latch state for an ATECC608 device.*
- **ATCA\_STATUS calib\_info\_get\_latch** (ATCADevice device, bool \*state)

*Use the Info command to get the persistent latch current state for an ATECC608 device.*
- **ATCA\_STATUS calib\_kdf** (ATCADevice device, uint8\_t mode, uint16\_t key\_id, const uint32\_t details, const uint8\_t \*message, uint8\_t \*out\_data, uint8\_t \*out\_nonce)

*Executes the KDF command, which derives a new key in PRF, AES, or HKDF modes.*
- **ATCA\_STATUS calib\_lock** (ATCADevice device, uint8\_t mode, uint16\_t summary\_crc)

*The Lock command prevents future modifications of the Configuration and/or Data and OTP zones. If the device is so configured, then this command can be used to lock individual data slots. This command fails if the designated area is already locked.*
- **ATCA\_STATUS calib\_lock\_config\_zone** (ATCADevice device)

- Unconditionally (no CRC required) lock the config zone.*

  - **ATCA\_STATUS calib\_lock\_config\_zone\_crc** (ATCADevice device, uint16\_t summary\_crc)

*Lock the config zone with summary CRC.*
- **ATCA\_STATUS calib\_lock\_data\_zone** (ATCADevice device)

*Unconditionally (no CRC required) lock the data zone (slots and OTP).*
- **ATCA\_STATUS calib\_lock\_data\_zone\_crc** (ATCADevice device, uint16\_t summary\_crc)

*Lock the data zone (slots and OTP) with summary CRC.*
- **ATCA\_STATUS calib\_lock\_data\_slot** (ATCADevice device, uint16\_t slot)

*Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (KeyConfig.Lockable=1).*
- **ATCA\_STATUS calib\_mac** (ATCADevice device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*challenge, uint8\_t \*digest)

*Executes MAC command, which computes a SHA-256 digest of a key stored in the device, a challenge, and other information on the device.*
- **ATCA\_STATUS calib\_nonce\_base** (ATCADevice device, uint8\_t mode, uint16\_t zero, const uint8\_t \*num\_in, uint8\_t \*rand\_out)

*Executes Nonce command, which loads a random or fixed nonce/data into the device for use by subsequent commands.*
- **ATCA\_STATUS calib\_nonce** (ATCADevice device, const uint8\_t \*num\_in)

*Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.*
- **ATCA\_STATUS calib\_nonce\_load** (ATCADevice device, uint8\_t target, const uint8\_t \*num\_in, uint16\_t num\_in\_size)

*Execute a Nonce command in pass-through mode to load one of the device's internal buffers with a fixed value.*
- **ATCA\_STATUS calib\_nonce\_rand** (ATCADevice device, const uint8\_t \*num\_in, uint8\_t \*rand\_out)

*Execute a Nonce command to generate a random nonce combining a host nonce (num\_in) and a device random number.*
- **ATCA\_STATUS calib\_challenge** (ATCADevice device, const uint8\_t \*num\_in)

*Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.*
- **ATCA\_STATUS calib\_challenge\_seed\_update** (ATCADevice device, const uint8\_t \*num\_in, uint8\_t \*rand\_out)

*Execute a Nonce command to generate a random challenge combining a host nonce (num\_in) and a device random number.*
- **ATCA\_STATUS calib\_priv\_write** (ATCADevice device, uint16\_t key\_id, const uint8\_t priv\_key[36], uint16\_t write\_key\_id, const uint8\_t write\_key[32], const uint8\_t num\_in[(20)])
- **ATCA\_STATUS calib\_random** (ATCADevice device, uint8\_t \*rand\_out)

*Executes Random command, which generates a 32 byte random number from the CryptoAuth device.*
- **ATCA\_STATUS calib\_read\_zone** (ATCADevice device, uint8\_t zone, uint16\_t slot, uint8\_t block, uint8\_t offset, uint8\_t \*data, uint8\_t len)

*Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.*
- **ATCA\_STATUS calib\_is\_locked** (ATCADevice device, uint8\_t zone, bool \*is\_locked)

*Executes Read command, which reads the configuration zone to see if the specified zone is locked.*
- **ATCA\_STATUS calib\_is\_slot\_locked** (ATCADevice device, uint16\_t slot, bool \*is\_locked)

*Executes Read command, which reads the configuration zone to see if the specified slot is locked.*
- **ATCA\_STATUS calib\_read\_bytes\_zone** (ATCADevice device, uint8\_t zone, uint16\_t slot, size\_t offset, uint8\_t \*data, size\_t length)

*Used to read an arbitrary number of bytes from any zone configured for clear reads.*
- **ATCA\_STATUS calib\_read\_serial\_number** (ATCADevice device, uint8\_t \*serial\_number)

*Executes Read command, which reads the 9 byte serial number of the device from the config zone.*
- **ATCA\_STATUS calib\_read\_pubkey** (ATCADevice device, uint16\_t slot, uint8\_t \*public\_key)

*Executes Read command to read an ECC P256 public key from a slot configured for clear reads.*
- **ATCA\_STATUS calib\_read\_sig** (ATCADevice device, uint16\_t slot, uint8\_t \*sig)

*Executes Read command to read a 64 byte ECDSA P256 signature from a slot configured for clear reads.*

- **ATCA\_STATUS calib\_read\_config\_zone** (ATCADevice device, uint8\_t \*config\_data)  
*Executes Read command to read the complete device configuration zone.*
- **ATCA\_STATUS calib\_cmp\_config\_zone** (ATCADevice device, uint8\_t \*config\_data, bool \*same\_config)  
*Compares a specified configuration zone with the configuration zone currently on the device.*
- **ATCA\_STATUS calib\_read\_enc** (ATCADevice device, uint16\_t key\_id, uint8\_t block, uint8\_t \*data, const uint8\_t \*enc\_key, const uint16\_t enc\_key\_id, const uint8\_t num\_in[(20)])
- **ATCA\_STATUS calib\_secureboot** (ATCADevice device, uint8\_t mode, uint16\_t param2, const uint8\_t \*digest, const uint8\_t \*signature, uint8\_t \*mac)  
*Executes Secure Boot command, which provides support for secure boot of an external MCU or MPU.*
- **ATCA\_STATUS calib\_secureboot\_mac** (ATCADevice device, uint8\_t mode, const uint8\_t \*digest, const uint8\_t \*signature, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)  
*Executes Secure Boot command with encrypted digest and validated MAC response using the IO protection key.*
- **ATCA\_STATUS calib\_selftest** (ATCADevice device, uint8\_t mode, uint16\_t param2, uint8\_t \*result)  
*Executes the SelfTest command, which performs a test of one or more of the cryptographic engines within the ATCC608 chip.*
- **ATCA\_STATUS calib\_sha\_base** (ATCADevice device, uint8\_t mode, uint16\_t length, const uint8\_t \*data\_in, uint8\_t \*data\_out, uint16\_t \*data\_out\_size)  
*Executes SHA command, which computes a SHA-256 or HMAC/SHA-256 digest for general purpose use by the host system.*
- **ATCA\_STATUS calib\_sha\_start** (ATCADevice device)  
*Executes SHA command to initialize SHA-256 calculation engine.*
- **ATCA\_STATUS calib\_sha\_update** (ATCADevice device, const uint8\_t \*message)  
*Executes SHA command to add 64 bytes of message data to the current context.*
- **ATCA\_STATUS calib\_sha\_end** (ATCADevice device, uint8\_t \*digest, uint16\_t length, const uint8\_t \*message)  
*Executes SHA command to complete SHA-256 or HMAC/SHA-256 operation.*
- **ATCA\_STATUS calib\_sha\_read\_context** (ATCADevice device, uint8\_t \*context, uint16\_t \*context\_size)  
*Executes SHA command to read the SHA-256 context back. Only for ATECC608 with SHA-256 contexts. HMAC not supported.*
- **ATCA\_STATUS calib\_sha\_write\_context** (ATCADevice device, const uint8\_t \*context, uint16\_t context\_size)  
*Executes SHA command to write (restore) a SHA-256 context into the device. Only supported for ATECC608 with SHA-256 contexts.*
- **ATCA\_STATUS calib\_sha** (ATCADevice device, uint16\_t length, const uint8\_t \*message, uint8\_t \*digest)  
*Use the SHA command to compute a SHA-256 digest.*
- **ATCA\_STATUS calib\_hw\_sha2\_256** (ATCADevice device, const uint8\_t \*data, size\_t data\_size, uint8\_t \*digest)  
*Use the SHA command to compute a SHA-256 digest.*
- **ATCA\_STATUS calib\_hw\_sha2\_256\_init** (ATCADevice device, atca\_sha256\_ctx\_t \*ctx)  
*Initialize a SHA context for performing a hardware SHA-256 operation on a device. Note that only one SHA operation can be run at a time.*
- **ATCA\_STATUS calib\_hw\_sha2\_256\_update** (ATCADevice device, atca\_sha256\_ctx\_t \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Add message data to a SHA context for performing a hardware SHA-256 operation on a device.*
- **ATCA\_STATUS calib\_hw\_sha2\_256\_finish** (ATCADevice device, atca\_sha256\_ctx\_t \*ctx, uint8\_t \*digest)  
*Finish SHA-256 digest for a SHA context for performing a hardware SHA-256 operation on a device.*
- **ATCA\_STATUS calib\_sha\_hmac\_init** (ATCADevice device, atca\_hmac\_sha256\_ctx\_t \*ctx, uint16\_t key\_slot)  
*Executes SHA command to start an HMAC/SHA-256 operation.*
- **ATCA\_STATUS calib\_sha\_hmac\_update** (ATCADevice device, atca\_hmac\_sha256\_ctx\_t \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Executes SHA command to add an arbitrary amount of message data to a HMAC/SHA-256 operation.*
- **ATCA\_STATUS calib\_sha\_hmac\_finish** (ATCADevice device, atca\_hmac\_sha256\_ctx\_t \*ctx, uint8\_t \*digest, uint8\_t target)



*Executes SHA command to complete a HMAC/SHA-256 operation.*

- **ATCA\_STATUS calib\_sha\_hmac** (ATCADevice device, const uint8\_t \*data, size\_t data\_size, uint16\_t key\_id, uint8\_t \*digest, uint8\_t target)

*Use the SHA command to compute an HMAC/SHA-256 operation.*

- **ATCA\_STATUS calib\_sign\_base** (ATCADevice device, uint8\_t mode, uint16\_t key\_id, uint8\_t \*signature)

*Executes the Sign command, which generates a signature using the ECDSA algorithm.*

- **ATCA\_STATUS calib\_sign** (ATCADevice device, uint16\_t key\_id, const uint8\_t \*msg, uint8\_t \*signature)

*Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*

- **ATCA\_STATUS calib\_sign\_internal** (ATCADevice device, uint16\_t key\_id, bool is\_invalidate, bool is\_full\_sn, uint8\_t \*signature)

*Executes Sign command to sign an internally generated message.*

- **ATCA\_STATUS calib\_updateextra** (ATCADevice device, uint8\_t mode, uint16\_t new\_value)

*Executes UpdateExtra command to update the values of the two extra bytes within the Configuration zone (bytes 84 and 85).*

- **ATCA\_STATUS calib\_verify** (ATCADevice device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*public\_key, const uint8\_t \*other\_data, uint8\_t \*mac)

*Executes the Verify command, which takes an ECDSA [R,S] signature and verifies that it is correctly generated from a given message and public key. In all cases, the signature is an input to the command.*

- **ATCA\_STATUS calib\_verify\_extern** (ATCADevice device, const uint8\_t \*message, const uint8\_t \*signature, const uint8\_t \*public\_key, bool \*is\_verified)

*Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*

- **ATCA\_STATUS calib\_verify\_extern\_mac** (ATCADevice device, const uint8\_t \*message, const uint8\_t \*signature, const uint8\_t \*public\_key, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)

*Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. This function is only available on the ATECC608.*

- **ATCA\_STATUS calib\_verify\_stored** (ATCADevice device, const uint8\_t \*message, const uint8\_t \*signature, uint16\_t key\_id, bool \*is\_verified)

*Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*

- **ATCA\_STATUS calib\_verify\_stored\_mac** (ATCADevice device, const uint8\_t \*message, const uint8\_t \*signature, uint16\_t key\_id, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)

*Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with a public key stored in the device. This function is only available on the ATECC608.*

- **ATCA\_STATUS calib\_verify\_validate** (ATCADevice device, uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*other\_data, bool \*is\_verified)

*Executes the Verify command in Validate mode to validate a public key stored in a slot.*

- **ATCA\_STATUS calib\_verify\_invalidate** (ATCADevice device, uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*other\_data, bool \*is\_verified)

*Executes the Verify command in Invalidate mode which invalidates a previously validated public key stored in a slot.*

- **ATCA\_STATUS calib\_write** (ATCADevice device, uint8\_t zone, uint16\_t address, const uint8\_t \*value, const uint8\_t \*mac)

*Executes the Write command, which writes either one four byte word or a 32-byte block to one of the EEPROM zones on the device. Depending upon the value of the WriteConfig byte for this slot, the data may be required to be encrypted by the system prior to being sent to the device. This command cannot be used to write slots configured as ECC private keys.*

- **ATCA\_STATUS calib\_write\_zone** (ATCADevice device, uint8\_t zone, uint16\_t slot, uint8\_t block, uint8\_t offset, const uint8\_t \*data, uint8\_t len)

*Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.*

- **ATCA\_STATUS calib\_write\_bytes\_zone** (ATCADevice device, uint8\_t zone, uint16\_t slot, size\_t offset\_bytes, const uint8\_t \*data, size\_t length)



*Executes the Write command, which writes data into the configuration, otp, or data zones with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).*

- `ATCA_STATUS_calib_write_pubkey` (`ATCADevice` device, `uint16_t` slot, `const uint8_t *`public\_key)

*Uses the write command to write a public key to a slot in the proper format.*

- `ATCA_STATUS_calib_write_config_zone` (`ATCADevice` device, `const uint8_t *`config\_data)

*Executes the Write command, which writes the configuration zone.*

- `ATCA_STATUS_calib_write_enc` (`ATCADevice` device, `uint16_t` key\_id, `uint8_t` block, `const uint8_t *`data, `const uint8_t *`enc\_key, `const uint16_t` enc\_key\_id, `const uint8_t` num\_in[(20)])
- `ATCA_STATUS_calib_write_config_counter` (`ATCADevice` device, `uint16_t` counter\_id, `uint32_t` counter\_↵value)

*Initialize one of the monotonic counters in device with a specific value.*

- `const char *` `atca_basic_aes_gcm_version` = "2.0"

## 18.7.1 Detailed Description

These methods provide a simple API to CryptoAuth chips.

## 18.7.2 Typedef Documentation

### 18.7.2.1 `atca_hmac_sha256_ctx_t`

```
typedef atca_sha256_ctx_t atca_hmac_sha256_ctx_t
```

### 18.7.2.2 `atca_sha256_ctx_t`

```
typedef struct atca_sha256_ctx atca_sha256_ctx_t
```

## 18.7.3 Function Documentation

### 18.7.3.1 `_calib_exit()`

```
ATCA_STATUS _calib_exit (
    ATCADevice device )
```

common cleanup code which idles the device after any operation

### Parameters

in	<i>device</i>	Device context pointer
----	---------------	------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.2 calib\_aes()

```
ATCA_STATUS calib_aes (
    ATCADevice device,
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * aes_in,
    uint8_t * aes_out )
```

Compute the AES-128 encrypt, decrypt, or GFM calculation.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	The mode for the AES command.
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>aes_in</i>	Input data to the AES command (16 bytes).
out	<i>aes_out</i>	Output data from the AES command is returned here (16 bytes).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.3 calib\_aes\_decrypt()

```
ATCA_STATUS calib_aes_decrypt (
    ATCADevice device,
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * ciphertext,
    uint8_t * plaintext )
```

Perform an AES-128 decrypt operation with a key in the device.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>ciphertext</i>	Input ciphertext to be decrypted (16 bytes).
out	<i>plaintext</i>	Output plaintext is returned here (16 bytes).

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.4 calib\_aes\_encrypt()**

```
ATCA_STATUS calib_aes_encrypt (
    ATCADevice device,
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * plaintext,
    uint8_t * ciphertext )
```

Perform an AES-128 encrypt operation with a key in the device.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>plaintext</i>	Input plaintext to be encrypted (16 bytes).
out	<i>ciphertext</i>	Output ciphertext is returned here (16 bytes).

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.5 calib\_aes\_gfm()**

```
ATCA_STATUS calib_aes_gfm (
    ATCADevice device,
    const uint8_t * h,
    const uint8_t * input,
    uint8_t * output )
```

Perform a Galois Field Multiply (GFM) operation.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>h</i>	First input value (16 bytes).
in	<i>input</i>	Second input value (16 bytes).
out	<i>output</i>	GFM result is returned here (16 bytes).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.6 calib\_cfg\_discover()

```
ATCA_STATUS calib_cfg_discover (
    ATCAInterfaceCfg cfg_array[],
    int max_ifaces )
```

auto discovery of crypto auth devices

Calls interface discovery functions and fills in `cfg_array` up to the maximum number of configurations either found or the size of the array. The `cfg_array` can have a mixture of interface types (ie: some I2C, some SWI or UART) depending upon which interfaces you've enabled

### Parameters

out	<i>cfg_array</i>	ptr to an array of interface configs
in	<i>max_ifaces</i>	maximum size of <code>cfg_array</code>

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.7 calib\_challenge()

```
ATCA_STATUS calib_challenge (
    ATCADevice device,
    const uint8_t * num_in )
```

Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>num_in</i>	Data to be loaded into TempKey (32 bytes).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.8 calib\_challenge\_seed\_update()

```
ATCA_STATUS calib_challenge_seed_update (
    ATCADevice device,
    const uint8_t * num_in,
    uint8_t * rand_out )
```

Execute a Nonce command to generate a random challenge combining a host nonce (num\_in) and a device random number.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>num_in</i>	Host nonce to be combined with the device random number (20 bytes).
out	<i>rand_out</i>	Internally generated 32-byte random number that was used in the nonce/challenge calculation is returned here. Can be NULL if not needed.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.9 calib\_checkmac()

```
ATCA_STATUS calib_checkmac (
    ATCADevice device,
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * challenge,
    const uint8_t * response,
    const uint8_t * other_data )
```

Compares a MAC response with input values.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Controls which fields within the device are used in the message
in	<i>key_id</i>	Key location in the CryptoAuth device to use for the MAC
in	<i>challenge</i>	Challenge data (32 bytes)
in	<i>response</i>	MAC response data (32 bytes)
in	<i>other_data</i>	OtherData parameter (13 bytes)

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.10 calib\_cmp\_config\_zone()

```
ATCA_STATUS calib_cmp_config_zone (
    ATCADevice device,
    uint8_t * config_data,
    bool * same_config )
```

Compares a specified configuration zone with the configuration zone currently on the device.

This only compares the static portions of the configuration zone and skips those that are unique per device (first 16 bytes) and areas that can change after the configuration zone has been locked (e.g. LastKeyUse).

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>config_data</i>	Full configuration data to compare the device against.
out	<i>same_config</i>	Result is returned here. True if the static portions on the configuration zones are the same.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

Max for all configs

### 18.7.3.11 calib\_counter()

```
ATCA_STATUS calib_counter (
    ATCADevice device,
    uint8_t mode,
    uint16_t counter_id,
    uint32_t * counter_value )
```

Compute the Counter functions.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	the mode used for the counter
in	<i>counter_id</i>	The counter to be used
out	<i>counter_value</i>	pointer to the counter value returned from device

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.12 calib\_counter\_increment()**

```
ATCA_STATUS calib_counter_increment (
    ATCADevice device,
    uint16_t counter_id,
    uint32_t * counter_value )
```

Increments one of the device's monotonic counters.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>counter_id</i>	Counter to be incremented
out	<i>counter_value</i>	New value of the counter is returned here. Can be NULL if not needed.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.13 calib\_counter\_read()**

```
ATCA_STATUS calib_counter_read (
    ATCADevice device,
    uint16_t counter_id,
    uint32_t * counter_value )
```

Read one of the device's monotonic counters.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>counter_id</i>	Counter to be read
out	<i>counter_value</i>	Counter value is returned here.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.14 calib\_derivekey()**

```
ATCA_STATUS calib_derivekey (
    ATCADevice device,
    uint8_t mode,
    uint16_t target_key,
    const uint8_t * mac )
```

Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Bit 2 must match the value in TempKey.SourceFlag
in	<i>target_key</i>	Key slot to be written
in	<i>mac</i>	Optional 32 byte MAC used to validate operation. NULL if not required.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.15 calib\_ecdh()

```
ATCA_STATUS calib_ecdh (
    ATCADevice device,
    uint16_t key_id,
    const uint8_t * public_key,
    uint8_t * pms )
```

ECDH command with a private key in a slot and the premaster secret is returned in the clear.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Slot of key for ECDH computation
in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	<i>pms</i>	Computed ECDH premaster secret is returned here. 32 bytes.

### Returns

ATCA\_SUCCESS on success

#### 18.7.3.16 calib\_ecdh\_base()

```
ATCA_STATUS calib_ecdh_base (
    ATCADevice device,
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * public_key,
    uint8_t * pms,
    uint8_t * out_nonce )
```

Base function for generating premaster secret key using ECDH.



## Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Mode to be used for ECDH computation
in	<i>key_id</i>	Slot of key for ECDH computation
in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	<i>pms</i>	Computed ECDH pre-master secret is returned here (32 bytes) if returned directly. Otherwise NULL.
out	<i>out_nonce</i>	Nonce used to encrypt pre-master secret. NULL if output encryption not used.

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 18.7.3.17 calib\_ecdh\_enc()

```
ATCA_STATUS calib_ecdh_enc (
    ATCADevice device,
    uint16_t key_id,
    const uint8_t * public_key,
    uint8_t * pms,
    const uint8_t * read_key,
    uint16_t read_key_id,
    const uint8_t num_in[ (20) ] )
```

## 18.7.3.18 calib\_ecdh\_ioenc()

```
ATCA_STATUS calib_ecdh_ioenc (
    ATCADevice device,
    uint16_t key_id,
    const uint8_t * public_key,
    uint8_t * pms,
    const uint8_t * io_key )
```

ECDH command with a private key in a slot and the premaster secret is returned encrypted using the IO protection key.

## Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Slot of key for ECDH computation
in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	<i>pms</i>	Computed ECDH premaster secret is returned here (32 bytes).
in	<i>io_key</i>	IO protection key.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.19 calib\_ecdh\_tempkey()

```
ATCA_STATUS calib_ecdh_tempkey (
    ATCADevice device,
    const uint8_t * public_key,
    uint8_t * pms )
```

ECDH command with a private key in TempKey and the premaster secret is returned in the clear.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	<i>pms</i>	Computed ECDH premaster secret is returned here (32 bytes).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.20 calib\_ecdh\_tempkey\_ioenc()

```
ATCA_STATUS calib_ecdh_tempkey_ioenc (
    ATCADevice device,
    const uint8_t * public_key,
    uint8_t * pms,
    const uint8_t * io_key )
```

ECDH command with a private key in TempKey and the premaster secret is returned encrypted using the IO protection key.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	<i>pms</i>	Computed ECDH premaster secret is returned here (32 bytes).
in	<i>io_key</i>	IO protection key.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.21 calib\_gendig()

```
ATCA_STATUS calib_gendig (
    ATCADevice device,
    uint8_t zone,
    uint16_t key_id,
    const uint8_t * other_data,
    uint8_t other_data_size )
```

Issues a GenDig command, which performs a SHA256 hash on the source data indicated by zone with the contents of TempKey. See the CryptoAuth datasheet for your chip to see what the values of zone correspond to.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>zone</i>	Designates the source of the data to hash with TempKey.
in	<i>key_id</i>	Indicates the key, OTP block, or message order for shared nonce mode.
in	<i>other_data</i>	Four bytes of data for SHA calculation when using a NoMac key, 32 bytes for "Shared Nonce" mode, otherwise ignored (can be NULL).
in	<i>other_data_size</i>	Size of other_data in bytes.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.22 calib\_genkey()

```
ATCA_STATUS calib_genkey (
    ATCADevice device,
    uint16_t key_id,
    uint8_t * public_key )
```

Issues GenKey command, which generates a new random private key in slot and returns the public key.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Slot number where an ECC private key is configured. Can also be ATCA_TEMPKEY_KEYID to generate a private key in TempKey.
out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Set to NULL if public key isn't required.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.23 calib\_genkey\_base()

```
ATCA_STATUS calib_genkey_base (
    ATCADevice device,
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * other_data,
    uint8_t * public_key )
```

Issues GenKey command, which can generate a private key, compute a public key, nd/or compute a digest of a public key.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Mode determines what operations the GenKey command performs.
in	<i>key_id</i>	Slot to perform the GenKey command on.
in	<i>other_data</i>	OtherData for PubKey digest calculation. Can be set to NULL otherwise.
out	<i>public_key</i>	If the mode indicates a public key will be calculated, it will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Set to NULL if public key isn't required.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.24 calib\_get\_addr()

```
ATCA_STATUS calib_get_addr (
    uint8_t zone,
    uint16_t slot,
    uint8_t block,
    uint8_t offset,
    uint16_t * addr )
```

Compute the address given the zone, slot, block, and offset.

#### Parameters

in	<i>zone</i>	Zone to get address from. Config(0), OTP(1), or Data(2) which requires a slot.
in	<i>slot</i>	Slot Id number for data zone and zero for other zones.
in	<i>block</i>	Block number within the data or configuration or OTP zone .
in	<i>offset</i>	Offset Number within the block of data or configuration or OTP zone.
out	<i>addr</i>	Pointer to the address of data or configuration or OTP zone.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.25 calib\_get\_pubkey()

```
ATCA_STATUS calib_get_pubkey (
    ATCADevice device,
    uint16_t key_id,
    uint8_t * public_key )
```

Uses GenKey command to calculate the public key from an existing private key in a slot.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Slot number of the private key.
out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve. Set to NULL if public key isn't required.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.26 calib\_get\_zone\_size()

```
ATCA_STATUS calib_get_zone_size (
    ATCADevice device,
    uint8_t zone,
    uint16_t slot,
    size_t * size )
```

Gets the size of the specified zone in bytes.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>zone</i>	Zone to get size information from. Config(0), OTP(1), or Data(2) which requires a slot.
in	<i>slot</i>	If zone is Data(2), the slot to query for size.
out	<i>size</i>	Zone size is returned here.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.27 calib\_hmac()

```
ATCA_STATUS calib_hmac (
    ATCADevice device,
```

## 18.7 Basic Crypto API methods for CryptoAuth Devices (calib\_)

```
uint8_t mode,  
uint16_t key_id,  
uint8_t * digest )
```

Issues a HMAC command, which computes an HMAC/SHA-256 digest of a key stored in the device, a challenge, and other information on the device.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Controls which fields within the device are used in the message.
in	<i>key_id</i>	Which key is to be used to generate the response. Bits 0:3 only are used to select a slot but all 16 bits are used in the HMAC message.
out	<i>digest</i>	HMAC digest is returned in this buffer (32 bytes).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.28 calib\_hw\_sha2\_256()

```
ATCA_STATUS calib_hw_sha2_256 (  
    ATCADevice device,  
    const uint8_t * data,  
    size_t data_size,  
    uint8_t * digest )
```

Use the SHA command to compute a SHA-256 digest.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>data</i>	Message data to be hashed.
in	<i>data_size</i>	Size of data in bytes.
out	<i>digest</i>	Digest is returned here (32 bytes).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.29 calib\_hw\_sha2\_256\_finish()

```
ATCA_STATUS calib_hw_sha2_256_finish (  
    ATCADevice device,  
    atca_sha256_ctx_t * ctx,  
    uint8_t * digest )
```

Finish SHA-256 digest for a SHA context for performing a hardware SHA-256 operation on a device.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	SHA256 context
out	<i>digest</i>	SHA256 digest is returned here (32 bytes)

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.30 calib\_hw\_sha2\_256\_init()**

```
ATCA_STATUS calib_hw_sha2_256_init (
    ATCADevice device,
    atca_sha256_ctx_t * ctx )
```

Initialize a SHA context for performing a hardware SHA-256 operation on a device. Note that only one SHA operation can be run at a time.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	SHA256 context

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.31 calib\_hw\_sha2\_256\_update()**

```
ATCA_STATUS calib_hw_sha2_256_update (
    ATCADevice device,
    atca_sha256_ctx_t * ctx,
    const uint8_t * data,
    size_t data_size )
```

Add message data to a SHA context for performing a hardware SHA-256 operation on a device.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	SHA256 context
in	<i>data</i>	Message data to be added to hash.
in	<i>data_size</i>	Size of data in bytes.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.32 calib\_idle()

```
ATCA_STATUS calib_idle (
    ATCADevice device )
```

idle the CryptoAuth device

### Parameters

in	<i>device</i>	Device context pointer
----	---------------	------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.33 calib\_info()

```
ATCA_STATUS calib_info (
    ATCADevice device,
    uint8_t * revision )
```

Use the Info command to get the device revision (DevRev).

### Parameters

in	<i>device</i>	Device context pointer
out	<i>revision</i>	Device revision is returned here (4 bytes).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.34 calib\_info\_base()

```
ATCA_STATUS calib_info_base (
    ATCADevice device,
    uint8_t mode,
    uint16_t param2,
    uint8_t * out_data )
```

Issues an Info command, which return internal device information and can control GPIO and the persistent latch.



**Parameters**

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Selects which mode to be used for info command.
in	<i>param2</i>	Selects the particular fields for the mode.
out	<i>out_data</i>	Response from info command (4 bytes). Can be set to NULL if not required.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.35 calib\_info\_get\_latch()**

```
ATCA_STATUS calib_info_get_latch (
    ATCADevice device,
    bool * state )
```

Use the Info command to get the persistent latch current state for an ATECC608 device.

**Parameters**

in	<i>device</i>	Device context pointer
out	<i>state</i>	The state is returned here. Set (true) or Cleared (false).

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.36 calib\_info\_set\_latch()**

```
ATCA_STATUS calib_info_set_latch (
    ATCADevice device,
    bool state )
```

Use the Info command to set the persistent latch state for an ATECC608 device.

**Parameters**

in	<i>device</i>	Device context pointer
out	<i>state</i>	Persistent latch state. Set (true) or clear (false).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.37 calib\_is\_locked()

```
ATCA_STATUS calib_is_locked (
    ATCADevice device,
    uint8_t zone,
    bool * is_locked )
```

Executes Read command, which reads the configuration zone to see if the specified zone is locked.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>zone</i>	The zone to query for locked (use LOCK_ZONE_CONFIG or LOCK_ZONE_DATA).
out	<i>is_locked</i>	Lock state returned here. True if locked.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.38 calib\_is\_slot\_locked()

```
ATCA_STATUS calib_is_slot_locked (
    ATCADevice device,
    uint16_t slot,
    bool * is_locked )
```

Executes Read command, which reads the configuration zone to see if the specified slot is locked.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>slot</i>	Slot to query for locked (slot 0-15)
out	<i>is_locked</i>	Lock state returned here. True if locked.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.39 calib\_kdf()

```
ATCA_STATUS calib_kdf (
    ATCADevice device,
    uint8_t mode,
    uint16_t key_id,
    const uint32_t details,
    const uint8_t * message,
    uint8_t * out_data,
    uint8_t * out_nonce )
```

Executes the KDF command, which derives a new key in PRF, AES, or HKDF modes.

Generally this function combines a source key with an input string and creates a result key/digest/array.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Mode determines KDF algorithm (PRF,AES,HKDF), source key location, and target key locations.
in	<i>key_id</i>	Source and target key slots if locations are in the EEPROM. Source key slot is the LSB and target key slot is the MSB.
in	<i>details</i>	Further information about the computation, depending on the algorithm (4 bytes).
in	<i>message</i>	Input value from system (up to 128 bytes). Actual size of message is 16 bytes for AES algorithm or is encoded in the MSB of the details parameter for other algorithms.
out	<i>out_data</i>	Output of the KDF function is returned here. If the result remains in the device, this can be NULL.
out	<i>out_nonce</i>	If the output is encrypted, a 32 byte random nonce generated by the device is returned here. If output encryption is not used, this can be NULL.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.40 calib\_lock()

```
ATCA_STATUS calib_lock (
    ATCADevice device,
    uint8_t mode,
    uint16_t summary_crc )
```

The Lock command prevents future modifications of the Configuration and/or Data and OTP zones. If the device is so configured, then this command can be used to lock individual data slots. This command fails if the designated area is already locked.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Zone, and/or slot, and summary check (bit 7).
in	<i>summary_crc</i>	CRC of the config or data zones. Ignored for slot locks or when mode bit 7 is set.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.41 calib\_lock\_config\_zone()

```
ATCA_STATUS calib_lock_config_zone (
    ATCADevice device )
```

Unconditionally (no CRC required) lock the config zone.

### Parameters

in	<i>device</i>	Device context pointer
----	---------------	------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.42 calib\_lock\_config\_zone\_crc()

```
ATCA_STATUS calib_lock_config_zone_crc (
    ATCADevice device,
    uint16_t summary_crc )
```

Lock the config zone with summary CRC.

The CRC is calculated over the entire config zone contents. 88 bytes for ATSHA devices, 128 bytes for ATECC devices. Lock will fail if the provided CRC doesn't match the internally calculated one.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>summary_crc</i>	Expected CRC over the config zone.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.43 calib\_lock\_data\_slot()

```
ATCA_STATUS calib_lock_data_slot (
    ATCADevice device,
    uint16_t slot )
```

Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (KeyConfig.Lockable=1).

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>slot</i>	Slot to be locked in data zone.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.44 calib\_lock\_data\_zone()

```
ATCA_STATUS calib_lock_data_zone (
    ATCADevice device )
```

Unconditionally (no CRC required) lock the data zone (slots and OTP).

ConfigZone must be locked and DataZone must be unlocked for the zone to be successfully locked.

#### Parameters

in	<i>device</i>	Device context pointer
----	---------------	------------------------

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.45 calib\_lock\_data\_zone\_crc()

```
ATCA_STATUS calib_lock_data_zone_crc (
    ATCADevice device,
    uint16_t summary_crc )
```

Lock the data zone (slots and OTP) with summary CRC.

The CRC is calculated over the concatenated contents of all the slots and OTP at the end. Private keys (KeyConfig.Private=1) are skipped. Lock will fail if the provided CRC doesn't match the internally calculated one.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>summary_crc</i>	Expected CRC over the data zone.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.46 calib\_mac()

```
ATCA_STATUS calib_mac (
    ATCADevice device,
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * challenge,
    uint8_t * digest )
```

Executes MAC command, which computes a SHA-256 digest of a key stored in the device, a challenge, and other information on the device.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Controls which fields within the device are used in the message
in	<i>key_id</i>	Key in the CryptoAuth device to use for the MAC
in	<i>challenge</i>	Challenge message (32 bytes). May be NULL if mode indicates a challenge isn't required.
out	<i>digest</i>	MAC response is returned here (32 bytes).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.47 calib\_nonce()

```
ATCA_STATUS calib_nonce (
    ATCADevice device,
    const uint8_t * num_in )
```

Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>num_in</i>	Data to be loaded into TempKey (32 bytes).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.48 calib\_nonce\_base()**

```

ATCA_STATUS calib_nonce_base (
    ATCADevice device,
    uint8_t mode,
    uint16_t zero,
    const uint8_t * num_in,
    uint8_t * rand_out )

```

Executes Nonce command, which loads a random or fixed nonce/data into the device for use by subsequent commands.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Controls the mechanism of the internal RNG or fixed write.
in	<i>zero</i>	Param2, normally 0, but can be used to indicate a nonce calculation mode (bit 15).
in	<i>num_in</i>	Input value to either be included in the nonce calculation in random modes (20 bytes) or to be written directly (32 bytes or 64 bytes(ATECC608)) in pass-through mode.
out	<i>rand_out</i>	If using a random mode, the internally generated 32-byte random number that was used in the nonce calculation is returned here. Can be NULL if not needed.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.49 calib\_nonce\_load()**

```

ATCA_STATUS calib_nonce_load (
    ATCADevice device,
    uint8_t target,
    const uint8_t * num_in,
    uint16_t num_in_size )

```

Execute a Nonce command in pass-through mode to load one of the device's internal buffers with a fixed value.

For the ATECC608, available targets are TempKey (32 or 64 bytes), Message Digest Buffer (32 or 64 bytes), or the Alternate Key Buffer (32 bytes). For all other devices, only TempKey (32 bytes) is available.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>target</i>	Target device buffer to load. Can be NONCE_MODE_TARGET_TEMPKEY, NONCE_MODE_TARGET_MSGDIGBUF, or NONCE_MODE_TARGET_ALTKEYBUF.
in	<i>num_in</i>	Data to load into the buffer.
in	<i>num_in_size</i>	Size of num_in in bytes. Can be 32 or 64 bytes depending on device and target.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.50 calib\_nonce\_rand()

```
ATCA_STATUS calib_nonce_rand (
    ATCADevice device,
    const uint8_t * num_in,
    uint8_t * rand_out )
```

Execute a Nonce command to generate a random nonce combining a host nonce (num\_in) and a device random number.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>num_in</i>	Host nonce to be combined with the device random number (20 bytes).
out	<i>rand_out</i>	Internally generated 32-byte random number that was used in the nonce/challenge calculation is returned here. Can be NULL if not needed.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.51 calib\_priv\_write()

```
ATCA_STATUS calib_priv_write (
    ATCADevice device,
    uint16_t key_id,
    const uint8_t priv_key[36],
    uint16_t write_key_id,
    const uint8_t write_key[32],
    const uint8_t num_in[(20)] )
```

#### 18.7.3.52 calib\_random()

```
ATCA_STATUS calib_random (
    ATCADevice device,
    uint8_t * rand_out )
```

Executes Random command, which generates a 32 byte random number from the CryptoAuth device.



## Parameters

in	<i>device</i>	Device context pointer
out	<i>rand_out</i>	32 bytes of random data is returned here.

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

18.7.3.53 **calib\_read\_bytes\_zone()**

```
ATCA_STATUS calib_read_bytes_zone (
    ATCADevice device,
    uint8_t zone,
    uint16_t slot,
    size_t offset,
    uint8_t * data,
    size_t length )
```

Used to read an arbitrary number of bytes from any zone configured for clear reads.

This function will issue the Read command as many times as is required to read the requested data.

## Parameters

in	<i>device</i>	Device context pointer
in	<i>zone</i>	Zone to read data from. Option are <a href="#">ATCA_ZONE_CONFIG(0)</a> , <a href="#">ATCA_ZONE_OTP(1)</a> , or <a href="#">ATCA_ZONE_DATA(2)</a> .
in	<i>slot</i>	Slot number to read from if zone is <a href="#">ATCA_ZONE_DATA(2)</a> . Ignored for all other zones.
in	<i>offset</i>	Byte offset within the zone to read from.
out	<i>data</i>	Read data is returned here.
in	<i>length</i>	Number of bytes to read starting from the offset.

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

18.7.3.54 **calib\_read\_config\_zone()**

```
ATCA_STATUS calib_read_config_zone (
    ATCADevice device,
    uint8_t * config_data )
```

Executes Read command to read the complete device configuration zone.

## 18.7 Basic Crypto API methods for CryptoAuth Devices (calib\_)

### Parameters

in	<i>device</i>	Device context pointer
out	<i>config_data</i>	Configuration zone data is returned here. 88 bytes for ATSHA devices, 128 bytes for ATECC devices.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.55 calib\_read\_enc()

```
ATCA_STATUS calib_read_enc (
    ATCADevice device,
    uint16_t key_id,
    uint8_t block,
    uint8_t * data,
    const uint8_t * enc_key,
    const uint16_t enc_key_id,
    const uint8_t num_in[ (20) ] )
```

#### 18.7.3.56 calib\_read\_pubkey()

```
ATCA_STATUS calib_read_pubkey (
    ATCADevice device,
    uint16_t slot,
    uint8_t * public_key )
```

Executes Read command to read an ECC P256 public key from a slot configured for clear reads.

This function assumes the public key is stored using the ECC public key format specified in the datasheet.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>slot</i>	Slot number to read from. Only slots 8 to 15 are large enough for a public key.
out	<i>public_key</i>	Public key is returned here (64 bytes). Format will be the 32 byte X and Y big-endian integers concatenated.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.57 calib\_read\_serial\_number()**

```
ATCA_STATUS calib_read_serial_number (
    ATCADevice device,
    uint8_t * serial_number )
```

Executes Read command, which reads the 9 byte serial number of the device from the config zone.

**Parameters**

in	<i>device</i>	Device context pointer
out	<i>serial_number</i>	9 byte serial number is returned here.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.58 calib\_read\_sig()**

```
ATCA_STATUS calib_read_sig (
    ATCADevice device,
    uint16_t slot,
    uint8_t * sig )
```

Executes Read command to read a 64 byte ECDSA P256 signature from a slot configured for clear reads.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>slot</i>	Slot number to read from. Only slots 8 to 15 are large enough for a signature.
out	<i>sig</i>	Signature will be returned here (64 bytes). Format will be the 32 byte R and S big-endian integers concatenated.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.59 calib\_read\_zone()**

```
ATCA_STATUS calib_read_zone (
    ATCADevice device,
    uint8_t zone,
    uint16_t slot,
    uint8_t block,
    uint8_t offset,
```

## 18.7 Basic Crypto API methods for CryptoAuth Devices (calib\_)

```
uint8_t * data,  
uint8_t len )
```

Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.

When reading a slot or OTP, data zone must be locked and the slot configuration must not be secret for a slot to be successfully read.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>zone</i>	Zone to be read from device. Options are ATCA_ZONE_CONFIG, ATCA_ZONE_OTP, or ATCA_ZONE_DATA.
in	<i>slot</i>	Slot number for data zone and ignored for other zones.
in	<i>block</i>	32 byte block index within the zone.
in	<i>offset</i>	4 byte work index within the block. Ignored for 32 byte reads.
out	<i>data</i>	Read data is returned here.
in	<i>len</i>	Length of the data to be read. Must be either 4 or 32.

returns ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.60 calib\_secureboot()

```
ATCA_STATUS calib_secureboot (  
    ATCADevice device,  
    uint8_t mode,  
    uint16_t param2,  
    const uint8_t * digest,  
    const uint8_t * signature,  
    uint8_t * mac )
```

Executes Secure Boot command, which provides support for secure boot of an external MCU or MPU.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Mode determines what operations the SecureBoot command performs.
in	<i>param2</i>	Not used, must be 0.
in	<i>digest</i>	Digest of the code to be verified (32 bytes).
in	<i>signature</i>	Signature of the code to be verified (64 bytes). Can be NULL when using the FullStore mode.
out	<i>mac</i>	Validating MAC will be returned here (32 bytes). Can be NULL if not required.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.61 calib\_secureboot\_mac()**

```

ATCA_STATUS calib_secureboot_mac (
    ATCADevice device,
    uint8_t mode,
    const uint8_t * digest,
    const uint8_t * signature,
    const uint8_t * num_in,
    const uint8_t * io_key,
    bool * is_verified )

```

Executes Secure Boot command with encrypted digest and validated MAC response using the IO protection key.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Mode determines what operations the SecureBoot command performs.
in	<i>digest</i>	Digest of the code to be verified (32 bytes). This is the plaintext digest (not encrypted).
in	<i>signature</i>	Signature of the code to be verified (64 bytes). Can be NULL when using the FullStore mode.
in	<i>num_in</i>	Host nonce (20 bytes).
in	<i>io_key</i>	IO protection key (32 bytes).
out	<i>is_verified</i>	Verify result is returned here.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.62 calib\_selftest()**

```

ATCA_STATUS calib_selftest (
    ATCADevice device,
    uint8_t mode,
    uint16_t param2,
    uint8_t * result )

```

Executes the SelfTest command, which performs a test of one or more of the cryptographic engines within the ATECC608 chip.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Functions to test. Can be a bit field combining any of the following: SELFTEST_MODE_RNG, SELFTEST_MODE_ECDSA_VERIFY, SELFTEST_MODE_ECDSA_SIGN, SELFTEST_MODE_ECDH, SELFTEST_MODE_AES, SELFTEST_MODE_SHA, SELFTEST_MODE_ALL.
in	<i>param2</i>	Currently unused, should be 0.
out	<i>result</i>	Results are returned here as a bit field.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.63 calib\_sha()

```
ATCA_STATUS calib_sha (
    ATCADevice device,
    uint16_t length,
    const uint8_t * message,
    uint8_t * digest )
```

Use the SHA command to compute a SHA-256 digest.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>length</i>	Size of message parameter in bytes.
in	<i>message</i>	Message data to be hashed.
out	<i>digest</i>	Digest is returned here (32 bytes).

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.64 calib\_sha\_base()

```
ATCA_STATUS calib_sha_base (
    ATCADevice device,
    uint8_t mode,
    uint16_t length,
    const uint8_t * message,
    uint8_t * data_out,
    uint16_t * data_out_size )
```

Executes SHA command, which computes a SHA-256 or HMAC/SHA-256 digest for general purpose use by the host system.

Only the Start(0) and Compute(1) modes are available for ATSHA devices.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	SHA command mode Start(0), Update/Compute(1), End(2), Public(3), HMACstart(4), HMACend(5), Read_Context(6), or Write_Context(7). Also message digest target location for the ATECC608.
in	<i>length</i>	Number of bytes in the message parameter or KeySlot for the HMAC key if Mode is HMACstart(4) or Public(3).

## Parameters

in	<i>message</i>	Message bytes to be hashed or Write_Context if restoring a context on the ATECC608. Can be NULL if not required by the mode.
out	<i>data_out</i>	Data returned by the command (digest or context).
in, out	<i>data_out_size</i>	As input, the size of the data_out buffer. As output, the number of bytes returned in data_out.

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 18.7.3.65 calib\_sha\_end()

```
ATCA_STATUS calib_sha_end (
    ATCADevice device,
    uint8_t * digest,
    uint16_t length,
    const uint8_t * message )
```

Executes SHA command to complete SHA-256 or HMAC/SHA-256 operation.

## Parameters

in	<i>device</i>	Device context pointer
out	<i>digest</i>	Digest from SHA-256 or HMAC/SHA-256 will be returned here (32 bytes).
in	<i>length</i>	Length of any remaining data to include in hash. Max 64 bytes.
in	<i>message</i>	Remaining data to include in hash. NULL if length is 0.

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 18.7.3.66 calib\_sha\_hmac()

```
ATCA_STATUS calib_sha_hmac (
    ATCADevice device,
    const uint8_t * data,
    size_t data_size,
    uint16_t key_slot,
    uint8_t * digest,
    uint8_t target )
```

Use the SHA command to compute an HMAC/SHA-256 operation.

## 18.7 Basic Crypto API methods for CryptoAuth Devices (calib\_)

### Parameters

in	<i>device</i>	Device context pointer
in	<i>data</i>	Message data to be hashed.
in	<i>data_size</i>	Size of data in bytes.
in	<i>key_slot</i>	Slot key id to use for the HMAC calculation
out	<i>digest</i>	Digest is returned here (32 bytes).
in	<i>target</i>	Where to save the digest internal to the device. For ATECC608, can be SHA_MODE_TARGET_TEMPKEY, SHA_MODE_TARGET_MSGDIGBUF, or SHA_MODE_TARGET_OUT_ONLY. For all other devices, SHA_MODE_TARGET_TEMPKEY is the only option.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.67 calib\_sha\_hmac\_finish()

```
ATCA_STATUS calib_sha_hmac_finish (
    ATCADevice device,
    atca_hmac_sha256_ctx_t * ctx,
    uint8_t * digest,
    uint8_t target )
```

Executes SHA command to complete a HMAC/SHA-256 operation.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	HMAC/SHA-256 context
out	<i>digest</i>	HMAC/SHA-256 result is returned here (32 bytes).
in	<i>target</i>	Where to save the digest internal to the device. For ATECC608, can be SHA_MODE_TARGET_TEMPKEY, SHA_MODE_TARGET_MSGDIGBUF, or SHA_MODE_TARGET_OUT_ONLY. For all other devices, SHA_MODE_TARGET_TEMPKEY is the only option.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.68 calib\_sha\_hmac\_init()

```
ATCA_STATUS calib_sha_hmac_init (
    ATCADevice device,
    atca_hmac_sha256_ctx_t * ctx,
    uint16_t key_slot )
```

Executes SHA command to start an HMAC/SHA-256 operation.



## Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	HMAC/SHA-256 context
in	<i>key_slot</i>	Slot key id to use for the HMAC calculation

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.69 calib\_sha\_hmac\_update()**

```
ATCA_STATUS calib_sha_hmac_update (
    ATCADevice device,
    atca_hmac_sha256_ctx_t * ctx,
    const uint8_t * data,
    size_t data_size )
```

Executes SHA command to add an arbitrary amount of message data to a HMAC/SHA-256 operation.

## Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	HMAC/SHA-256 context
in	<i>data</i>	Message data to add
in	<i>data_size</i>	Size of message data in bytes

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.70 calib\_sha\_read\_context()**

```
ATCA_STATUS calib_sha_read_context (
    ATCADevice device,
    uint8_t * context,
    uint16_t * context_size )
```

Executes SHA command to read the SHA-256 context back. Only for ATECC608 with SHA-256 contexts. HMAC not supported.

## Parameters

in	<i>device</i>	Device context pointer
out	<i>context</i>	Context data is returned here.
in, out	<i>context_size</i>	As input, the size of the context buffer in bytes. As output, the size of the returned context data.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.71 calib\_sha\_start()

```
ATCA_STATUS calib_sha_start (
    ATCADevice device )
```

Executes SHA command to initialize SHA-256 calculation engine.

### Parameters

in	<i>device</i>	Device context pointer
----	---------------	------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.72 calib\_sha\_update()

```
ATCA_STATUS calib_sha_update (
    ATCADevice device,
    const uint8_t * message )
```

Executes SHA command to add 64 bytes of message data to the current context.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>message</i>	64 bytes of message data to add to add to operation.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.73 calib\_sha\_write\_context()

```
ATCA_STATUS calib_sha_write_context (
    ATCADevice device,
    const uint8_t * context,
    uint16_t context_size )
```

Executes SHA command to write (restore) a SHA-256 context into the the device. Only supported for ATECC608 with SHA-256 contexts.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>context</i>	Context data to be restored.
in	<i>context_size</i>	Size of the context data in bytes.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.74 calib\_sign()**

```
ATCA_STATUS calib_sign (
    ATCADevice device,
    uint16_t key_id,
    const uint8_t * msg,
    uint8_t * signature )
```

Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Slot of the private key to be used to sign the message.
in	<i>msg</i>	32-byte message to be signed. Typically the SHA256 hash of the full message.
out	<i>signature</i>	Signature will be returned here. Format is R and S integers in big-endian format. 64 bytes for P256 curve.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.75 calib\_sign\_base()**

```
ATCA_STATUS calib_sign_base (
    ATCADevice device,
    uint8_t mode,
    uint16_t key_id,
    uint8_t * signature )
```

Executes the Sign command, which generates a signature using the ECDSA algorithm.

## 18.7 Basic Crypto API methods for CryptoAuth Devices (calib\_)

### Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Mode determines what the source of the message to be signed.
in	<i>key_id</i>	Private key slot used to sign the message.
out	<i>signature</i>	Signature is returned here. Format is R and S integers in big-endian format. 64 bytes for P256 curve.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.76 calib\_sign\_internal()

```
ATCA_STATUS calib_sign_internal (
    ATCADevice device,
    uint16_t key_id,
    bool is_invalidate,
    bool is_full_sn,
    uint8_t * signature )
```

Executes Sign command to sign an internally generated message.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Slot of the private key to be used to sign the message.
in	<i>is_invalidate</i>	Set to true if the signature will be used with the Verify(Invalidate) command. false for all other cases.
in	<i>is_full_sn</i>	Set to true if the message should incorporate the device's full serial number.
out	<i>signature</i>	Signature is returned here. Format is R and S integers in big-endian format. 64 bytes for P256 curve.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.77 calib\_sleep()

```
ATCA_STATUS calib_sleep (
    ATCADevice device )
```

invoke sleep on the CryptoAuth device

## Parameters

in	<i>device</i>	Device context pointer
----	---------------	------------------------

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.78 calib\_updateextra()**

```
ATCA_STATUS calib_updateextra (
    ATCADevice device,
    uint8_t mode,
    uint16_t new_value )
```

Executes UpdateExtra command to update the values of the two extra bytes within the Configuration zone (bytes 84 and 85).

Can also be used to decrement the limited use counter associated with the key in slot NewValue.

## Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Mode determines what operations the UpdateExtra command performs.
in	<i>new_value</i>	Value to be written.

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.79 calib\_verify()**

```
ATCA_STATUS calib_verify (
    ATCADevice device,
    uint8_t mode,
    uint16_t key_id,
    const uint8_t * signature,
    const uint8_t * public_key,
    const uint8_t * other_data,
    uint8_t * mac )
```

Executes the Verify command, which takes an ECDSA [R,S] signature and verifies that it is correctly generated from a given message and public key. In all cases, the signature is an input to the command.

For the Stored, External, and ValidateExternal Modes, the contents of TempKey (or Message Digest Buffer in some cases for the ATECC608) should contain the 32 byte message.

## 18.7 Basic Crypto API methods for CryptoAuth Devices (calib\_)

### Parameters

in	<i>device</i>	Device context pointer
in	<i>mode</i>	Verify command mode and options
in	<i>key_id</i>	Stored mode, the slot containing the public key to be used for the verification. ValidateExternal mode, the slot containing the public key to be validated. External mode, KeyID contains the curve type to be used to Verify the signature. Validate or Invalidate mode, the slot containing the public key to be (in)validated.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>public_key</i>	If mode is External, the public key to be used for verification. X and Y integers in big-endian format. 64 bytes for P256 curve. NULL for all other modes.
in	<i>other_data</i>	If mode is Validate, the bytes used to generate the message for the validation (19 bytes). NULL for all other modes.
out	<i>mac</i>	If mode indicates a validating MAC, then the MAC will be returned here. Can be NULL otherwise.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.7.3.80 calib\_verify\_extern()

```
ATCA_STATUS calib_verify_extern (
    ATCADevice device,
    const uint8_t * message,
    const uint8_t * signature,
    const uint8_t * public_key,
    bool * is_verified )
```

Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>message</i>	32 byte message to be verified. Typically the SHA256 hash of the full message.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>public_key</i>	The public key to be used for verification. X and Y integers in big-endian format. 64 bytes for P256 curve.
out	<i>is_verified</i>	Boolean whether or not the message, signature, public key verified.

### Returns

ATCA\_SUCCESS on verification success or failure, because the command still completed successfully.

### 18.7.3.81 calib\_verify\_extern\_mac()

```
ATCA_STATUS calib_verify_extern_mac (
    ATCADevice device,
    const uint8_t * message,
    const uint8_t * signature,
    const uint8_t * public_key,
    const uint8_t * num_in,
    const uint8_t * io_key,
    bool * is_verified )
```

Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. This function is only available on the ATECC608.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>message</i>	32 byte message to be verified. Typically the SHA256 hash of the full message.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>public_key</i>	The public key to be used for verification. X and Y integers in big-endian format. 64 bytes for P256 curve.
in	<i>num_in</i>	System nonce (32 byte) used for the verification MAC.
in	<i>io_key</i>	IO protection key for verifying the validation MAC.
out	<i>is_verified</i>	Boolean whether or not the message, signature, public key verified.

#### Returns

ATCA\_SUCCESS on verification success or failure, because the command still completed successfully.

### 18.7.3.82 calib\_verify\_invalidate()

```
ATCA_STATUS calib_verify_invalidate (
    ATCADevice device,
    uint16_t key_id,
    const uint8_t * signature,
    const uint8_t * other_data,
    bool * is_verified )
```

Executes the Verify command in Invalidate mode which invalidates a previously validated public key stored in a slot.

This command can only be run after GenKey has been used to create a PubKey digest of the public key to be invalidated in TempKey (mode=0x10).

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Slot containing the public key to be invalidated.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>other_data</i>	19 bytes of data used to build the verification message.
out	<i>is_verified</i>	Boolean whether or not the message, signature, validation public key verified.

### Returns

ATCA\_SUCCESS on verification success or failure, because the command still completed successfully.

#### 18.7.3.83 calib\_verify\_stored()

```
ATCA_STATUS calib_verify_stored (
    ATCADevice device,
    const uint8_t * message,
    const uint8_t * signature,
    uint16_t key_id,
    bool * is_verified )
```

Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>message</i>	32 byte message to be verified. Typically the SHA256 hash of the full message.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>key_id</i>	Slot containing the public key to be used in the verification.
out	<i>is_verified</i>	Boolean whether or not the message, signature, public key verified.

### Returns

ATCA\_SUCCESS on verification success or failure, because the command still completed successfully.

#### 18.7.3.84 calib\_verify\_stored\_mac()

```
ATCA_STATUS calib_verify_stored_mac (
    ATCADevice device,
    const uint8_t * message,
    const uint8_t * signature,
    uint16_t key_id,
    const uint8_t * num_in,
    const uint8_t * io_key,
    bool * is_verified )
```

Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with a public key stored in the device. This function is only available on the ATECC608.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>message</i>	32 byte message to be verified. Typically the SHA256 hash of the full message.



**Parameters**

in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>key_id</i>	Slot containing the public key to be used in the verification.
in	<i>num_in</i>	System nonce (32 byte) used for the verification MAC.
in	<i>io_key</i>	IO protection key for verifying the validation MAC.
out	<i>is_verified</i>	Boolean whether or not the message, signature, public key verified.

**Returns**

ATCA\_SUCCESS on verification success or failure, because the command still completed successfully.

**18.7.3.85 calib\_verify\_validate()**

```
ATCA_STATUS calib_verify_validate (
    ATCADevice device,
    uint16_t key_id,
    const uint8_t * signature,
    const uint8_t * other_data,
    bool * is_verified )
```

Executes the Verify command in Validate mode to validate a public key stored in a slot.

This command can only be run after GenKey has been used to create a PubKey digest of the public key to be validated in TempKey (mode=0x10).

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Slot containing the public key to be validated.
in	<i>signature</i>	Signature to be verified. R and S integers in big-endian format. 64 bytes for P256 curve.
in	<i>other_data</i>	19 bytes of data used to build the verification message.
out	<i>is_verified</i>	Boolean whether or not the message, signature, validation public key verified.

**Returns**

ATCA\_SUCCESS on verification success or failure, because the command still completed successfully.

**18.7.3.86 calib\_wakeup()**

```
ATCA_STATUS calib_wakeup (
    ATCADevice device )
```

basic API methods are all prefixed with atcab\_ (CryptoAuthLib Basic) the fundamental premise of the basic API is it is based on a single interface instance and that instance is global, so all basic API commands assume that one global device is the one to operate on.

wakeup the CryptoAuth device

### Parameters

in	<i>device</i>	Device context pointer
----	---------------	------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.87 calib\_write()

```
ATCA_STATUS calib_write (
    ATCADevice device,
    uint8_t zone,
    uint16_t address,
    const uint8_t * value,
    const uint8_t * mac )
```

Executes the Write command, which writes either one four byte word or a 32-byte block to one of the EEPROM zones on the device. Depending upon the value of the WriteConfig byte for this slot, the data may be required to be encrypted by the system prior to being sent to the device. This command cannot be used to write slots configured as ECC private keys.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>zone</i>	Zone/Param1 for the write command.
in	<i>address</i>	Address/Param2 for the write command.
in	<i>value</i>	Plain-text data to be written or cipher-text for encrypted writes. 32 or 4 bytes depending on bit 7 in the zone.
in	<i>mac</i>	MAC required for encrypted writes (32 bytes). Set to NULL if not required.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.88 calib\_write\_bytes\_zone()

```
ATCA_STATUS calib_write_bytes_zone (
    ATCADevice device,
    uint8_t zone,
    uint16_t slot,
    size_t offset_bytes,
    const uint8_t * data,
    size_t length )
```

Executes the Write command, which writes data into the configuration, otp, or data zones with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).

Config zone must be unlocked for writes to that zone. If data zone is unlocked, only 32-byte writes are allowed to slots and OTP and the offset and length must be multiples of 32 or the write will fail.

## Parameters

in	<i>device</i>	Device context pointer
in	<i>zone</i>	Zone to write data to: <a href="#">ATCA_ZONE_CONFIG(0)</a> , <a href="#">ATCA_ZONE_OTP(1)</a> , or <a href="#">ATCA_ZONE_DATA(2)</a> .
in	<i>slot</i>	If zone is <a href="#">ATCA_ZONE_DATA(2)</a> , the slot number to write to. Ignored for all other zones.
in	<i>offset_bytes</i>	Byte offset within the zone to write to. Must be a multiple of a word (4 bytes).
in	<i>data</i>	Data to be written.
in	<i>length</i>	Number of bytes to be written. Must be a multiple of a word (4 bytes).

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.89 calib\_write\_config\_counter()**

```
ATCA_STATUS calib_write_config_counter (
    ATCADevice device,
    uint16_t counter_id,
    uint32_t counter_value )
```

Initialize one of the monotonic counters in device with a specific value.

The monotonic counters are stored in the configuration zone using a special format. This encodes a binary count value into the 8 byte encoded value required. Can only be set while the configuration zone is unlocked.

## Parameters

in	<i>device</i>	Device context pointer
in	<i>counter_id</i>	Counter to be written.
in	<i>counter_value</i>	Counter value to set.

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.7.3.90 calib\_write\_config\_zone()**

```
ATCA_STATUS calib_write_config_zone (
    ATCADevice device,
    const uint8_t * config_data )
```

Executes the Write command, which writes the configuration zone.

First 16 bytes are skipped as they are not writable. LockValue and LockConfig are also skipped and can only be changed via the Lock command.

This command may fail if UserExtra and/or Selector bytes have already been set to non-zero values.

## 18.7 Basic Crypto API methods for CryptoAuth Devices (calib\_)

### Parameters

in	<i>device</i>	Device context pointer
in	<i>config_data</i>	Data to the config zone data. This should be 88 bytes for SHA devices and 128 bytes for ECC devices.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.91 calib\_write\_enc()

```
ATCA_STATUS calib_write_enc (
    ATCADevice device,
    uint16_t key_id,
    uint8_t block,
    const uint8_t * data,
    const uint8_t * enc_key,
    const uint16_t enc_key_id,
    const uint8_t num_in[ (20) ] )
```

#### 18.7.3.92 calib\_write\_pubkey()

```
ATCA_STATUS calib_write_pubkey (
    ATCADevice device,
    uint16_t slot,
    const uint8_t * public_key )
```

Uses the write command to write a public key to a slot in the proper format.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>slot</i>	Slot number to write. Only slots 8 to 15 are large enough to store a public key.
in	<i>public_key</i>	Public key to write into the slot specified. X and Y integers in big-endian format. 64 bytes for P256 curve.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.7.3.93 calib\_write\_zone()

```
ATCA_STATUS calib_write_zone (
    ATCADevice device,
```

```

uint8_t zone,
uint16_t slot,
uint8_t block,
uint8_t offset,
const uint8_t * data,
uint8_t len )

```

Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>zone</i>	Device zone to write to (0=config, 1=OTP, 2=data).
in	<i>slot</i>	If writing to the data zone, it is the slot to write to, otherwise it should be 0.
in	<i>block</i>	32-byte block to write to.
in	<i>offset</i>	4-byte word within the specified block to write to. If performing a 32-byte write, this should be 0.
in	<i>data</i>	Data to be written.
in	<i>len</i>	Number of bytes to be written. Must be either 4 or 32.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 18.7.4 Variable Documentation

### 18.7.4.1 atca\_basic\_aes\_gcm\_version

```
const char* atca_basic_aes_gcm_version = "2.0"
```

## 18.8 Software crypto methods (atcac\_)

These methods provide a software implementation of various crypto algorithms.

### Macros

- #define `ATCA_ECC_P256_FIELD_SIZE` (256 / 8)
- #define `ATCA_ECC_P256_PRIVATE_KEY_SIZE` (`ATCA_ECC_P256_FIELD_SIZE`)
- #define `ATCA_ECC_P256_PUBLIC_KEY_SIZE` (`ATCA_ECC_P256_FIELD_SIZE` \* 2)
- #define `ATCA_ECC_P256_SIGNATURE_SIZE` (`ATCA_ECC_P256_FIELD_SIZE` \* 2)

### Functions

- int `atcac_sw_ecdsa_verify_p256` (const uint8\_t msg[(256/8)], const uint8\_t signature[((256/8) \* 2)], const uint8\_t public\_key[((256/8) \* 2)])  
*return software generated ECDSA verification result and the function is currently not implemented*
- int `atcac_sw_random` (uint8\_t \*data, size\_t data\_size)  
*return software generated random number and the function is currently not implemented*
- int `atcac_sw_sha1_init` (`atcac_sha1_ctx` \*ctx)  
*Initialize context for performing SHA1 hash in software.*
- int `atcac_sw_sha1_update` (`atcac_sha1_ctx` \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Add data to a SHA1 hash.*
- int `atcac_sw_sha1_finish` (`atcac_sha1_ctx` \*ctx, uint8\_t digest[(20)])
- int `atcac_sw_sha1` (const uint8\_t \*data, size\_t data\_size, uint8\_t digest[(20)])  
*Perform SHA1 hash of data in software.*
- int `atcac_sw_sha2_256_init` (`atcac_sha2_256_ctx` \*ctx)  
*Initialize context for performing SHA256 hash in software.*
- int `atcac_sw_sha2_256_update` (`atcac_sha2_256_ctx` \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Add data to a SHA256 hash.*
- int `atcac_sw_sha2_256_finish` (`atcac_sha2_256_ctx` \*ctx, uint8\_t digest[(32)])
- int `atcac_sw_sha2_256` (const uint8\_t \*data, size\_t data\_size, uint8\_t digest[(32)])  
*single call convenience function which computes Hash of given data using SHA256 software*
- `ATCA_STATUS` `atcac_sha256_hmac_init` (`atcac_hmac_sha256_ctx` \*ctx, const uint8\_t \*key, const uint8\_t key\_len)  
*Initialize context for performing HMAC (sha256) in software.*
- `ATCA_STATUS` `atcac_sha256_hmac_update` (`atcac_hmac_sha256_ctx` \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Update HMAC context with input data.*
- `ATCA_STATUS` `atcac_sha256_hmac_finish` (`atcac_hmac_sha256_ctx` \*ctx, uint8\_t \*digest, size\_t \*digest\_len)  
*Finish CMAC calculation and clear the HMAC context.*

### 18.8.1 Detailed Description

These methods provide a software implementation of various crypto algorithms.

### 18.8.2 Macro Definition Documentation

### 18.8.2.1 ATCA\_ECC\_P256\_FIELD\_SIZE

```
#define ATCA_ECC_P256_FIELD_SIZE (256 / 8)
```

### 18.8.2.2 ATCA\_ECC\_P256\_PRIVATE\_KEY\_SIZE

```
#define ATCA_ECC_P256_PRIVATE_KEY_SIZE (ATCA_ECC_P256_FIELD_SIZE)
```

### 18.8.2.3 ATCA\_ECC\_P256\_PUBLIC\_KEY\_SIZE

```
#define ATCA_ECC_P256_PUBLIC_KEY_SIZE (ATCA_ECC_P256_FIELD_SIZE * 2)
```

### 18.8.2.4 ATCA\_ECC\_P256\_SIGNATURE\_SIZE

```
#define ATCA_ECC_P256_SIGNATURE_SIZE (ATCA_ECC_P256_FIELD_SIZE * 2)
```

## 18.8.3 Function Documentation

### 18.8.3.1 atcac\_sha256\_hmac\_finish()

```
ATCA_STATUS atcac_sha256_hmac_finish (
    atcac_hmac_sha256_ctx * ctx,
    uint8_t * digest,
    size_t * digest_len )
```

Finish CMAC calculation and clear the HMAC context.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.8.3.2 atcac\_sha256\_hmac\_init()

```
ATCA_STATUS atcac_sha256_hmac_init (
    atcac_hmac_sha256_ctx * ctx,
    const uint8_t * key,
    const uint8_t key_len )
```

Initialize context for performing HMAC (sha256) in software.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.8.3.3 atcac\_sha256\_hmac\_update()

```
ATCA_STATUS atcac_sha256_hmac_update (
    atcac_hmac_sha256_ctx * ctx,
    const uint8_t * data,
    size_t data_size )
```

Update HMAC context with input data.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.8.3.4 atcac\_sw\_ecdsa\_verify\_p256()

```
int atcac_sw_ecdsa_verify_p256 (
    const uint8_t msg[(256/8)],
    const uint8_t signature[((256/8) *2)],
    const uint8_t public_key[((256/8) *2)] )
```

return software generated ECDSA verification result and the function is currently not implemented

#### Parameters

in	<i>msg</i>	ptr to message or challenge
in	<i>signature</i>	ptr to the signature to verify
in	<i>public_key</i>	ptr to public key of device which signed the challenge return ATCA_UNIMPLEMENTED , as the function is currently not implemented



**18.8.3.5 atcac\_sw\_random()**

```
int atcac_sw_random (
    uint8_t * data,
    size_t data_size )
```

return software generated random number and the function is currently not implemented

**Parameters**

out	<i>data</i>	ptr to space to receive the random number
in	<i>data_size</i>	size of data buffer return ATCA_UNIMPLEMENTED , as the function is not implemented

**18.8.3.6 atcac\_sw\_sha1()**

```
int atcac_sw_shal (
    const uint8_t * data,
    size_t data_size,
    uint8_t digest[(20)] )
```

Perform SHA1 hash of data in software.

**Parameters**

in	<i>data</i>	Data to be hashed
in	<i>data_size</i>	Data size in bytes
out	<i>digest</i>	Digest is returned here (20 bytes)

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.8.3.7 atcac\_sw\_sha1\_finish()**

```
int atcac_sw_shal_finish (
    atcac_shal_ctx * ctx,
    uint8_t digest[(20)] )
```

**18.8.3.8 atcac\_sw\_sha1\_init()**

```
int atcac_sw_shal_init (
    atcac_shal_ctx * ctx )
```

Initialize context for performing SHA1 hash in software.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.8.3.9 atcac\_sw\_sha1\_update()

```
int atcac_sw_sha1_update (
    atcac_sha1_ctx * ctx,
    const uint8_t * data,
    size_t data_size )
```

Add data to a SHA1 hash.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.8.3.10 atcac\_sw\_sha2\_256()

```
int atcac_sw_sha2_256 (
    const uint8_t * data,
    size_t data_size,
    uint8_t digest[(32)] )
```

single call convenience function which computes Hash of given data using SHA256 software

### Parameters

in	<i>data</i>	pointer to stream of data to hash
in	<i>data_size</i>	size of data stream to hash
out	<i>digest</i>	result

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.8.3.11 atcac\_sw\_sha2\_256\_finish()

```
int atcac_sw_sha2_256_finish (
    atcac_sha2_256_ctx * ctx,
    uint8_t digest[(32)] )
```

**18.8.3.12 atcac\_sw\_sha2\_256\_init()**

```
int atcac_sw_sha2_256_init (
    atcac_sha2_256_ctx * ctx )
```

Initialize context for performing SHA256 hash in software.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.8.3.13 atcac\_sw\_sha2\_256\_update()**

```
int atcac_sw_sha2_256_update (
    atcac_sha2_256_ctx * ctx,
    const uint8_t * data,
    size_t data_size )
```

Add data to a SHA256 hash.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

## 18.9 Hardware abstraction layer (hal\_)

These methods define the hardware abstraction layer for communicating with a CryptoAuth device.

### Data Structures

- struct [ATCAHAL\\_t](#)  
*an intermediary data structure to allow the HAL layer to point the standard API functions used by the upper layers to the HAL implementation for the interface. This isolates the upper layers and loosely couples the ATCAIface object from the physical implementation.*
- struct [atcahid](#)
- struct [i2c\\_start\\_instance](#)
- struct [atcal2Cmaster](#)  
*this is the hal\_data for ATCA HAL for ASF SERCOM*
- struct [hid\\_device](#)
- struct [atcaSPImaster](#)
- struct [i2c\\_sam\\_instance](#)
- struct [atcaSWImaster](#)  
*this is the hal\_data for ATCA HAL for ASF SERCOM*

### Macros

- #define [ATCA\\_POLLING\\_INIT\\_TIME\\_MSEC](#) 1
- #define [ATCA\\_POLLING\\_FREQUENCY\\_TIME\\_MSEC](#) 2
- #define [ATCA\\_POLLING\\_MAX\\_TIME\\_MSEC](#) 2500
- #define [hal\\_memset\\_s](#) [atcab\\_memset\\_s](#)
- #define [HID\\_DEVICES\\_MAX](#) 10
- #define [HID\\_PACKET\\_MAX](#) 512
- #define [MAX\\_I2C\\_BUSES](#) 2
- #define [HID\\_DEVICES\\_MAX](#) 10
- #define [HID\\_PACKET\\_MAX](#) 512
- #define [MAX\\_SPI\\_BUSES](#) 2
- #define [SWI\\_WAKE\\_TOKEN](#) ((uint8\_t)0x00)  
*flag preceding a command*
- #define [SWI\\_FLAG\\_CMD](#) ((uint8\_t)0x77)  
*flag preceding a command*
- #define [SWI\\_FLAG\\_TX](#) ((uint8\_t)0x88)  
*flag requesting a response*
- #define [SWI\\_FLAG\\_IDLE](#) ((uint8\_t)0xBB)  
*flag requesting to go into Idle mode*
- #define [SWI\\_FLAG\\_SLEEP](#) ((uint8\_t)0xCC)  
*flag requesting to go into Sleep mode*
- #define [MAX\\_I2C\\_BUSES](#) 3
- #define [HID\\_GUID](#) { 0x4d1e55b2, 0xf16f, 0x11cf, 0x88, 0xcb, 0x00, 0x11, 0x11, 0x00, 0x00, 0x30 }
- #define [HID\\_DEVICES\\_MAX](#) 10
- #define [HID\\_PACKET\\_MAX](#) 512
- #define [KIT\\_MAX\\_SCAN\\_COUNT](#) 4
- #define [KIT\\_MAX\\_TX\\_BUF](#) 32
- #define [KIT\\_TX\\_WRAP\\_SIZE](#) (10)
- #define [KIT\\_MSG\\_SIZE](#) (32)
- #define [KIT\\_RX\\_WRAP\\_SIZE](#) (KIT\_MSG\_SIZE + 6)

- `#define MAX_SWI_BUSES 6`
- `#define RECEIVE_MODE 0`
- `#define TRANSMIT_MODE 1`
- `#define RX_DELAY 10`
- `#define TX_DELAY 90`
- `#define DEBUG_PIN_1 EXT2_PIN_5`
- `#define DEBUG_PIN_2 EXT2_PIN_6`
- `#define MAX_SWI_BUSES 6`
- `#define RECEIVE_MODE 0`
- `#define TRANSMIT_MODE 1`
- `#define RX_DELAY 10`
- `#define TX_DELAY 93`

## Typedefs

- `typedef struct atcahid atcahid_t`
- `typedef void(* start_change_baudrate) (ATCAIface iface, uint32_t speed)`
- `typedef struct i2c_start_instance i2c_start_instance_t`
- `typedef struct atcal2Cmaster ATCAI2CMaster_t`
- `typedef struct hid_device hid_device_t`
- `typedef struct atcahid atcahid_t`
- `typedef struct atcaSPImaster ATCASPIMaster_t`
- `typedef void(* sam_change_baudrate) (ATCAIface iface, uint32_t speed)`
- `typedef struct i2c_sam_instance i2c_sam_instance_t`
- `typedef struct atcal2Cmaster ATCAI2CMaster_t`  
*this is the hal\_data for ATCA HAL for ASF SERCOM*
- `typedef struct hid_device hid_device_t`
- `typedef struct atcahid atcahid_t`
- `typedef struct atcaSWImaster ATCASWIMaster_t`  
*this is the hal\_data for ATCA HAL for ASF SERCOM*
- `typedef struct atcaSWImaster ATCASWIMaster_t`  
*this is the hal\_data for ATCA HAL for ASF SERCOM*

## Functions

- `ATCA_STATUS hal_iface_init (ATCAIfaceCfg *, ATCAHAL_t **hal)`  
*Standard HAL API for ATCA to initialize a physical interface.*
- `ATCA_STATUS hal_iface_release (ATCAIfaceType, void *hal_data)`  
*releases a physical interface, HAL knows how to interpret hal\_data*
- `ATCA_STATUS hal_check_wake (const uint8_t *response, int response_size)`  
*Utility function for hal\_wake to check the reply.*
- `void atca_delay_ms (uint32_t ms)`  
*Timer API for legacy implementations.*
- `void atca_delay_us (uint32_t delay)`  
*This function delays for a number of microseconds.*
- `void hal_rtos_delay_ms (uint32_t ms)`  
*Timer API implemented at the HAL level.*
- `void hal_delay_ms (uint32_t delay)`  
*This function delays for a number of milliseconds.*
- `void hal_delay_us (uint32_t delay)`  
*This function delays for a number of microseconds.*

- [ATCA\\_STATUS hal\\_create\\_mutex](#) (void \*\*ppMutex, char \*pName)  
*Optional hal interfaces.*
- [ATCA\\_STATUS hal\\_destroy\\_mutex](#) (void \*pMutex)
- [ATCA\\_STATUS hal\\_lock\\_mutex](#) (void \*pMutex)
- [ATCA\\_STATUS hal\\_unlock\\_mutex](#) (void \*pMutex)
- void \* [hal\\_malloc](#) (size\_t size)
- void [hal\\_free](#) (void \*ptr)
- [ATCA\\_STATUS hal\\_iface\\_register\\_hal](#) (ATCAIfaceType iface\_type, ATCAHAL\_t \*hal, ATCAHAL\_t \*\*old)  
*Register/Replace a HAL with a.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_discover\\_buses](#) (int hid\_buses[], int max\_buses)  
*discover cdc buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_discover\\_devices](#) (int bus\_num, ATCAIfaceCfg cfg[], int \*found)  
*discover any CryptoAuth devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_init](#) (void \*hal, ATCAIfaceCfg \*cfg)  
*HAL implementation of Kit USB HID init.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_post\\_init](#) (ATCAIface iface)  
*HAL implementation of Kit HID post init.*
- [ATCA\\_STATUS kit\\_phy\\_send](#) (ATCAIface iface, uint8\_t \*txdata, int txlength)  
*HAL implementation of send over USB HID.*
- [ATCA\\_STATUS kit\\_phy\\_receive](#) (ATCAIface iface, uint8\_t \*rxdata, int \*rxsize)  
*HAL implementation of kit protocol send over USB HID.*
- [ATCA\\_STATUS kit\\_phy\\_num\\_found](#) (int8\_t \*num\_found)  
*Number of USB HID devices found.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_send](#) (ATCAIface iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of kit protocol send over USB HID.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_receive](#) (ATCAIface iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxsize)  
*HAL implementation of send over USB HID.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_wake](#) (ATCAIface iface)  
*Call the wake for kit protocol.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_idle](#) (ATCAIface iface)  
*Call the idle for kit protocol.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_sleep](#) (ATCAIface iface)  
*Call the sleep for kit protocol.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_release](#) (void \*hal\_data)  
*Close the physical port for HID.*
- [ATCA\\_STATUS hal\\_i2c\\_discover\\_buses](#) (int i2c\_buses[], int max\_buses)  
*discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge*
- [ATCA\\_STATUS hal\\_i2c\\_discover\\_devices](#) (int bus\_num, ATCAIfaceCfg cfg[], int \*found)  
*discover any CryptoAuth devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_i2c\\_init](#) (void \*hal, ATCAIfaceCfg \*cfg)  
*hal\_i2c\_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIface instances using the same bus, and you can have multiple ATCAIface instances on multiple i2c buses, so hal\_i2c\_init manages these things and ATCAIface is abstracted from the physical details.*
- [ATCA\\_STATUS hal\\_i2c\\_post\\_init](#) (ATCAIface iface)  
*HAL implementation of I2C post init.*
- [ATCA\\_STATUS hal\\_i2c\\_send](#) (ATCAIface iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of I2C send over START.*
- [ATCA\\_STATUS hal\\_i2c\\_receive](#) (ATCAIface iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)

- HAL implementation of I2C receive function for START I2C.*
- void [change\\_i2c\\_speed](#) ([ATCAIface](#) iface, uint32\_t speed)
  - method to change the bus speed of I2C*
- [ATCA\\_STATUS hal\\_i2c\\_wake](#) ([ATCAIface](#) iface)
  - wake up CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_idle](#) ([ATCAIface](#) iface)
  - idle CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_sleep](#) ([ATCAIface](#) iface)
  - sleep CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_release](#) (void \*hal\_data)
  - manages reference count on given bus and releases resource if no more references exist*
- void [hal\\_delay\\_10us](#) (uint32\_t delay)
  - This function delays for a number of tens of microseconds.*
- void [atca\\_delay\\_10us](#) (uint32\_t delay)
  - This function delays for a number of tens of microseconds.*
- [ATCA\\_STATUS hal\\_spi\\_discover\\_buses](#) (int spi\_buses[], int max\_buses)
  - discover spi buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge*
- [ATCA\\_STATUS hal\\_spi\\_discover\\_devices](#) (int bus\_num, [ATCAIfaceCfg](#) cfg[], int \*found)
  - discover any TA100 devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_spi\\_init](#) (void \*hal, [ATCAIfaceCfg](#) \*cfg)
  - initialize an SPI interface using given config*
- [ATCA\\_STATUS hal\\_spi\\_post\\_init](#) ([ATCAIface](#) iface)
  - HAL implementation of SPI post init.*
- [ATCA\\_STATUS hal\\_spi\\_send](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)
  - HAL implementation of SPI send over Harmony.*
- [ATCA\\_STATUS hal\\_spi\\_receive](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)
  - HAL implementation of SPI receive function for HARMONY SPI.*
- [ATCA\\_STATUS hal\\_spi\\_wake](#) ([ATCAIface](#) iface)
  - wake up TA100 device using SPI bus*
- [ATCA\\_STATUS hal\\_spi\\_idle](#) ([ATCAIface](#) iface)
  - idle TA100 device using SPI bus*
- [ATCA\\_STATUS hal\\_spi\\_sleep](#) ([ATCAIface](#) iface)
  - sleep TA100 device using SPI bus*
- [ATCA\\_STATUS hal\\_spi\\_release](#) (void \*hal\_data)
  - manages reference count on given bus and releases resource if no more references exist*
- [ATCA\\_STATUS hal\\_swi\\_discover\\_buses](#) (int swi\_buses[], int max\_buses)
  - discover swi buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge*
- [ATCA\\_STATUS hal\\_swi\\_discover\\_devices](#) (int bus\_num, [ATCAIfaceCfg](#) cfg[], int \*found)
  - discover any CryptoAuth devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_swi\\_init](#) (void \*hal, [ATCAIfaceCfg](#) \*cfg)
  - hal\_swi\_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIFace instances using the same bus, and you can have multiple ATCAIFace instances on multiple swi buses, so hal\_swi\_init manages these things and ATCAIFace is abstracted from the physical details.*
- [ATCA\\_STATUS hal\\_swi\\_post\\_init](#) ([ATCAIface](#) iface)
  - HAL implementation of SWI post init.*
- [ATCA\\_STATUS hal\\_swi\\_send\\_flag](#) ([ATCAIface](#) iface, uint8\_t data)
  - HAL implementation of SWI send one byte over UART.*
- [ATCA\\_STATUS hal\\_swi\\_send](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)
  - HAL implementation of SWI send command over UART.*

- **ATCA\_STATUS hal\_swi\_receive** (ATCAIface iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)  
*HAL implementation of SWI receive function over UART.*
- **ATCA\_STATUS hal\_swi\_wake** (ATCAIface iface)  
*wake up CryptoAuth device using SWI interface*
- **ATCA\_STATUS hal\_swi\_idle** (ATCAIface iface)  
*idle CryptoAuth device using SWI interface*
- **ATCA\_STATUS hal\_swi\_sleep** (ATCAIface iface)  
*sleep CryptoAuth device using SWI interface*
- **ATCA\_STATUS hal\_swi\_release** (void \*hal\_data)  
*manages reference count on given bus and releases resource if no more references exist*
- **ATCA\_STATUS kit\_phy\_send** (ATCAIface iface, const char \*txdata, int txlength)  
*HAL implementation of kit protocol send .It is called by the top layer.*
- **ATCA\_STATUS kit\_phy\_receive** (ATCAIface iface, char \*rxdata, int rxsize)  
*HAL implementation of kit protocol receive data.It is called by the top layer.*
- **char \* strnchr** (const char \*s, size\_t count, int c)
- **const char \* kit\_id\_from\_devtype** (ATCADeviceType devtype)
- **const char \* kit\_interface\_from\_kitype** (ATCAKitType kitype)
- **ATCA\_STATUS kit\_init** (ATCAIface iface)  
*HAL implementation of kit protocol init. This function calls back to the physical protocol to send the bytes.*
- **ATCA\_STATUS kit\_send** (ATCAIface iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of kit protocol send. This function calls back to the physical protocol to send the bytes.*
- **ATCA\_STATUS kit\_receive** (ATCAIface iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxsize)  
*HAL implementation to receive bytes and unwrap from kit protocol. This function calls back to the physical protocol to receive the bytes.*
- **ATCA\_STATUS kit\_wake** (ATCAIface iface)  
*Call the wake for kit protocol.*
- **ATCA\_STATUS kit\_idle** (ATCAIface iface)  
*Call the idle for kit protocol.*
- **ATCA\_STATUS kit\_sleep** (ATCAIface iface)  
*Call the sleep for kit protocol.*
- **ATCA\_STATUS kit\_wrap\_cmd** (const uint8\_t \*txdata, int txlen, char \*pkitcmd, int \*nkitcmd, char target)  
*Wrap binary bytes in ascii kit protocol.*
- **ATCA\_STATUS kit\_parse\_rsp** (const char \*pkitbuf, int nkitbuf, uint8\_t \*kitstatus, uint8\_t \*rxdata, int \*datasize)  
*Parse the response ascii from the kit.*
- **ATCA\_STATUS swi\_uart\_init** (ATCASWIMaster\_t \*instance)  
*Implementation of SWI UART init.*
- **ATCA\_STATUS swi\_uart\_deinit** (ATCASWIMaster\_t \*instance)  
*Implementation of SWI UART deinit.*
- **void swi\_uart\_setbaud** (ATCASWIMaster\_t \*instance, uint32\_t baudrate)  
*implementation of SWI UART change baudrate.*
- **void swi\_uart\_mode** (ATCASWIMaster\_t \*instance, uint8\_t mode)  
*implementation of SWI UART change mode.*
- **void swi\_uart\_discover\_buses** (int swi\_uart\_buses[], int max\_buses)  
*discover UART buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge*
- **ATCA\_STATUS swi\_uart\_send\_byte** (ATCASWIMaster\_t \*instance, uint8\_t data)  
*HAL implementation of SWI UART send byte over ASF. This function send one byte over UART.*
- **ATCA\_STATUS swi\_uart\_receive\_byte** (ATCASWIMaster\_t \*instance, uint8\_t \*data)  
*HAL implementation of SWI UART receive bytes over ASF. This function receive one byte over UART.*



## Variables

- [atcahid\\_t\\_gHid](#)
- [atcahid\\_t\\_gHid](#)
- [atcahid\\_t\\_gHid](#)
- struct port\_config [pin\\_conf](#)

### 18.9.1 Detailed Description

These methods define the hardware abstraction layer for communicating with a CryptoAuth device.

These methods define the hardware abstraction layer for communicating with a CryptoAuth device using SWI interface.

These methods define the hardware abstraction layer for communicating with a CryptoAuth device using SWI Interface.

These methods define the hardware abstraction layer for communicating with a TA100 device.

< Uncomment when debugging

These methods define the hardware abstraction layer for communicating with a CryptoAuth device using I2C driver of ASF.

### 18.9.2 Macro Definition Documentation

#### 18.9.2.1 ATCA\_POLLING\_FREQUENCY\_TIME\_MSEC

```
#define ATCA_POLLING_FREQUENCY_TIME_MSEC 2
```

#### 18.9.2.2 ATCA\_POLLING\_INIT\_TIME\_MSEC

```
#define ATCA_POLLING_INIT_TIME_MSEC 1
```

#### 18.9.2.3 ATCA\_POLLING\_MAX\_TIME\_MSEC

```
#define ATCA_POLLING_MAX_TIME_MSEC 2500
```

### 18.9.2.4 DEBUG\_PIN\_1

```
#define DEBUG_PIN_1 EXT2_PIN_5
```

### 18.9.2.5 DEBUG\_PIN\_2

```
#define DEBUG_PIN_2 EXT2_PIN_6
```

### 18.9.2.6 hal\_memset\_s

```
#define hal_memset_s atcab_memset_s
```

### 18.9.2.7 HID\_DEVICES\_MAX [1/3]

```
#define HID_DEVICES_MAX 10
```

### 18.9.2.8 HID\_DEVICES\_MAX [2/3]

```
#define HID_DEVICES_MAX 10
```

### 18.9.2.9 HID\_DEVICES\_MAX [3/3]

```
#define HID_DEVICES_MAX 10
```

### 18.9.2.10 HID\_GUID

```
#define HID_GUID { 0x4d1e55b2, 0xf16f, 0x11cf, 0x88, 0xcb, 0x00, 0x11, 0x11, 0x00, 0x00, 0x30  
}
```

**18.9.2.11 HID\_PACKET\_MAX [1/3]**

```
#define HID_PACKET_MAX 512
```

**18.9.2.12 HID\_PACKET\_MAX [2/3]**

```
#define HID_PACKET_MAX 512
```

**18.9.2.13 HID\_PACKET\_MAX [3/3]**

```
#define HID_PACKET_MAX 512
```

**18.9.2.14 KIT\_MAX\_SCAN\_COUNT**

```
#define KIT_MAX_SCAN_COUNT 4
```

**18.9.2.15 KIT\_MAX\_TX\_BUF**

```
#define KIT_MAX_TX_BUF 32
```

**18.9.2.16 KIT\_MSG\_SIZE**

```
#define KIT_MSG_SIZE (32)
```

**18.9.2.17 KIT\_RX\_WRAP\_SIZE**

```
#define KIT_RX_WRAP_SIZE (KIT_MSG_SIZE + 6)
```

**18.9.2.18 KIT\_TX\_WRAP\_SIZE**

```
#define KIT_TX_WRAP_SIZE (10)
```

### 18.9.2.19 MAX\_I2C\_BUSES [1/2]

```
#define MAX_I2C_BUSES 2
```

### 18.9.2.20 MAX\_I2C\_BUSES [2/2]

```
#define MAX_I2C_BUSES 3
```

### 18.9.2.21 MAX\_SPI\_BUSES

```
#define MAX_SPI_BUSES 2
```

### 18.9.2.22 MAX\_SWI\_BUSES [1/2]

```
#define MAX_SWI_BUSES 6
```

- this HAL implementation assumes you've included the ASF SERCOM UART libraries in your project, otherwise, the HAL layer will not compile because the ASF UART drivers are a dependency \*

### 18.9.2.23 MAX\_SWI\_BUSES [2/2]

```
#define MAX_SWI_BUSES 6
```

- this HAL implementation assumes you've included the ASF SERCOM UART libraries in your project, otherwise, the HAL layer will not compile because the ASF UART drivers are a dependency \*

### 18.9.2.24 RECEIVE\_MODE [1/2]

```
#define RECEIVE_MODE 0
```

**18.9.2.25 RECEIVE\_MODE [2/2]**

```
#define RECEIVE_MODE 0
```

**18.9.2.26 RX\_DELAY [1/2]**

```
#define RX_DELAY 10
```

**18.9.2.27 RX\_DELAY [2/2]**

```
#define RX_DELAY 10
```

**18.9.2.28 SWI\_FLAG\_CMD**

```
#define SWI_FLAG_CMD ((uint8_t)0x77)
```

flag preceding a command

**18.9.2.29 SWI\_FLAG\_IDLE**

```
#define SWI_FLAG_IDLE ((uint8_t)0xBB)
```

flag requesting to go into Idle mode

**18.9.2.30 SWI\_FLAG\_SLEEP**

```
#define SWI_FLAG_SLEEP ((uint8_t)0xCC)
```

flag requesting to go into Sleep mode

**18.9.2.31 SWI\_FLAG\_TX**

```
#define SWI_FLAG_TX ((uint8_t)0x88)
```

flag requesting a response

### 18.9.2.32 SWI\_WAKE\_TOKEN

```
#define SWI_WAKE_TOKEN ((uint8_t)0x00)
```

flag preceding a command

### 18.9.2.33 TRANSMIT\_MODE [1/2]

```
#define TRANSMIT_MODE 1
```

### 18.9.2.34 TRANSMIT\_MODE [2/2]

```
#define TRANSMIT_MODE 1
```

### 18.9.2.35 TX\_DELAY [1/2]

```
#define TX_DELAY 90
```

### 18.9.2.36 TX\_DELAY [2/2]

```
#define TX_DELAY 93
```

## 18.9.3 Typedef Documentation

### 18.9.3.1 atcahid\_t [1/3]

```
typedef struct atcahid atcahid_t
```

### 18.9.3.2 atcahid\_t [2/3]

```
typedef struct atcahid atcahid_t
```

**18.9.3.3 atcahid\_t [3/3]**

```
typedef struct atcahid atcahid_t
```

**18.9.3.4 ATCAI2CMaster\_t [1/2]**

```
typedef struct atcaI2Cmaster ATCAI2CMaster_t
```

**18.9.3.5 ATCAI2CMaster\_t [2/2]**

```
typedef struct atcaI2Cmaster ATCAI2CMaster_t
```

this is the hal\_data for ATCA HAL for ASF SERCOM

**18.9.3.6 ATCASPIMaster\_t**

```
typedef struct atcaSPImaster ATCASPIMaster_t
```

**18.9.3.7 ATCASWIMaster\_t [1/2]**

```
typedef struct atcaSWImaster ATCASWIMaster_t
```

this is the hal\_data for ATCA HAL for ASF SERCOM

**18.9.3.8 ATCASWIMaster\_t [2/2]**

```
typedef struct atcaSWImaster ATCASWIMaster_t
```

this is the hal\_data for ATCA HAL for ASF SERCOM

**18.9.3.9 hid\_device\_t [1/2]**

```
typedef struct hid_device hid_device_t
```

### 18.9.3.10 hid\_device\_t [2/2]

```
typedef struct hid_device hid_device_t
```

### 18.9.3.11 i2c\_sam\_instance\_t

```
typedef struct i2c_sam_instance i2c_sam_instance_t
```

### 18.9.3.12 i2c\_start\_instance\_t

```
typedef struct i2c_start_instance i2c_start_instance_t
```

### 18.9.3.13 sam\_change\_baudrate

```
typedef void(* sam_change_baudrate) (ATCAIface iface, uint32_t speed)
```

### 18.9.3.14 start\_change\_baudrate

```
typedef void(* start_change_baudrate) (ATCAIface iface, uint32_t speed)
```

## 18.9.4 Function Documentation

### 18.9.4.1 atca\_delay\_10us()

```
void atca_delay_10us (  
    uint32_t delay )
```

This function delays for a number of tens of microseconds.

#### Parameters

in	<i>delay</i>	number of 0.01 milliseconds to delay
----	--------------	--------------------------------------



#### 18.9.4.2 atca\_delay\_ms()

```
void atca_delay_ms (
    uint32_t delay )
```

Timer API for legacy implementations.

This function delays for a number of milliseconds.

You can override this function if you like to do something else in your system while delaying.

##### Parameters

in	<i>delay</i>	number of milliseconds to delay
----	--------------	---------------------------------

#### 18.9.4.3 atca\_delay\_us()

```
void atca_delay_us (
    uint32_t delay )
```

This function delays for a number of microseconds.

##### Parameters

in	<i>delay</i>	number of 0.001 milliseconds to delay
in	<i>delay</i>	number of microseconds to delay

#### 18.9.4.4 change\_i2c\_speed()

```
ATCA_STATUS change_i2c_speed (
    ATCAIface iface,
    uint32_t speed )
```

method to change the bus speec of I2C

method to change the bus speed of I2C

method to change the bus speed of I2C.This function is not used in Linux.

##### Parameters

in	<i>iface</i>	interface on which to change bus speed
in	<i>speed</i>	baud rate (typically 100000 or 400000)
in	<i>iface</i>	interface on which to change bus speed
in	<i>speed</i>	baud rate (typically 100000 or 400000)

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.9.4.5 hal\_check\_wake()

```
ATCA_STATUS hal_check_wake (
    const uint8_t * response,
    int response_size )
```

Utility function for hal\_wake to check the reply.

### Parameters

in	<i>response</i>	Wake response to be checked.
in	<i>response_size</i>	Size of the response to check.

### Returns

ATCA\_SUCCESS for expected wake, ATCA\_STATUS\_SELFTEST\_ERROR if the power on self test failed, ATCA\_WAKE\_FAILED for other failures.

#### 18.9.4.6 hal\_create\_mutex()

```
ATCA_STATUS hal_create_mutex (
    void ** ppMutex,
    char * pName )
```

Optional hal interfaces.

Application callback for creating a mutex object.

### Parameters

in, out	<i>ppMutex</i>	location to receive ptr to mutex
in, out	<i>pName</i>	String used to identify the mutex
	<i>[IN/OUT]</i>	ppMutex location to receive ptr to mutex
	<i>[IN]</i>	pName Name of the mutex for systems using named objects

#### 18.9.4.7 hal\_delay\_10us()

```
void hal_delay_10us (
    uint32_t delay )
```

This function delays for a number of tens of microseconds.

**Parameters**

in	<i>delay</i>	number of 0.01 milliseconds to delay
----	--------------	--------------------------------------

**18.9.4.8 hal\_delay\_ms()**

```
void hal_delay_ms (
    uint32_t delay )
```

This function delays for a number of milliseconds.

You can override this function if you like to do something else in your system while delaying.

**Parameters**

in	<i>delay</i>	number of milliseconds to delay
----	--------------	---------------------------------

**18.9.4.9 hal\_delay\_us()**

```
void hal_delay_us (
    uint32_t delay )
```

This function delays for a number of microseconds.

**Parameters**

in	<i>delay</i>	number of microseconds to delay
----	--------------	---------------------------------

**18.9.4.10 hal\_destroy\_mutex()**

```
ATCA_STATUS hal_destroy_mutex (
    void * pMutex )
```

**18.9.4.11 hal\_free()**

```
void hal_free (
    void * ptr )
```

### 18.9.4.12 hal\_i2c\_discover\_buses()

```
ATCA_STATUS hal_i2c_discover_buses (
    int i2c_buses[],
    int max_buses )
```

discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge

This HAL implementation assumes you've included the ASF TWI libraries in your project, otherwise, the HAL layer will not compile because the ASF TWI drivers are a dependency.

logical to physical bus mapping structure

discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge. This function is not implemented.

discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge

#### Parameters

in	<i>i2c_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover

#### Returns

ATCA\_SUCCESS

#### Parameters

in	<i>i2c_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover

#### Returns

ATCA\_UNIMPLEMENTED

discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge

#### Parameters

in	<i>i2c_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover

#### Returns

ATCA\_SUCCESS

discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge

## Parameters

in	<i>i2c_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover return ATCA_SUCCESS

**18.9.4.13 hal\_i2c\_discover\_devices()**

```
ATCA_STATUS hal_i2c_discover_devices (
    int bus_num,
    ATCAIfaceCfg cfg[],
    int * found )
```

discover any CryptoAuth devices on a given logical bus number

## Parameters

in	<i>bus_num</i>	logical bus number on which to look for CryptoAuth devices
out	<i>cfg</i>	pointer to head of an array of interface config structures which get filled in by this method
out	<i>found</i>	number of devices found on this bus

## Returns

ATCA\_SUCCESS

## Parameters

in	<i>bus_num</i>	logical bus number on which to look for CryptoAuth devices
out	<i>cfg</i>	pointer to head of an array of interface config structures which get filled in by this method
out	<i>found</i>	number of devices found on this bus

## Returns

ATCA\_UNIMPLEMENTED

## Parameters

in	<i>bus_num</i>	- logical bus number on which to look for CryptoAuth devices
out	<i>cfg[]</i>	- pointer to head of an array of interface config structures which get filled in by this method
out	<i>*found</i>	- number of devices found on this bus

## Returns

ATCA\_SUCCESS

### Parameters

in	<i>bus_num</i>	Logical bus number on which to look for CryptoAuth devices
out	<i>cfg</i>	Pointer to head of an array of interface config structures which get filled in by this method
out	<i>found</i>	Number of devices found on this bus

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.9.4.14 hal\_i2c\_idle()

```
ATCA_STATUS hal_i2c_idle (
    ATCAIFace iface )
```

idle CryptoAuth device using I2C bus

### Parameters

in	<i>iface</i>	interface to logical device to idle
----	--------------	-------------------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.9.4.15 hal\_i2c\_init()

```
ATCA_STATUS hal_i2c_init (
    void * hal,
    ATCAIFaceCfg * cfg )
```

hal\_i2c\_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIFace instances using the same bus, and you can have multiple ATCAIFace instances on multiple i2c buses, so hal\_i2c\_init manages these things and ATCAIFace is abstracted from the physical details.

hal\_i2c\_init manages requests to initialize a physical interface. It manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIFace instances using the same bus, and you can have multiple ATCAIFace instances on multiple i2c buses, so hal\_i2c\_init manages these things and ATCAIFace is abstracted from the physical details.

initialize an I2C interface using given config

HAL implementation of I2C init.

- this HAL implementation assumes you've included the START Twi libraries in your project, otherwise, the HAL layer will not compile because the START TWI drivers are a dependency \*

initialize an I2C interface using given config

**Parameters**

in	<i>hal</i>	- opaque ptr to HAL data
in	<i>cfg</i>	- interface configuration

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

this implementation assumes I2C peripheral has been enabled by user. It only initialize an I2C interface using given config.

**Parameters**

in	<i>hal</i>	pointer to HAL specific data that is maintained by this HAL
in	<i>cfg</i>	pointer to HAL specific configuration data that is used to initialize this HAL

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

- this HAL implementation assumes you've included the ASF SERCOM I2C libraries in your project, otherwise, the HAL layer will not compile because the ASF I2C drivers are a dependency \*

**Parameters**

in	<i>hal</i>	- opaque ptr to HAL data
in	<i>cfg</i>	- interface configuration

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

initialize an I2C interface using given config

**Parameters**

in	<i>hal</i>	- opaque ptr to HAL data
in	<i>cfg</i>	- interface configuration

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

- this HAL implementation assumes you've included the ASF Twi libraries in your project, otherwise, the HAL layer will not compile because the ASF TWI drivers are a dependency \*

initialize an I2C interface using given config

### Parameters

in	<i>hal</i>	- opaque ptr to HAL data
in	<i>cfg</i>	- interface configuration

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.9.4.16 hal\_i2c\_post\_init()

```
ATCA_STATUS hal_i2c_post_init (  
    ATCAIface iface )
```

HAL implementation of I2C post init.

### Parameters

in	<i>iface</i>	instance
----	--------------	----------

### Returns

ATCA\_SUCCESS

### Parameters

in	<i>iface</i>	instance
----	--------------	----------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.9.4.17 hal\_i2c\_receive()

```
ATCA_STATUS hal_i2c_receive (  
    ATCAIface iface,  
    uint8_t word_address,  
    uint8_t * rxdata,  
    uint16_t * rxlength )
```

HAL implementation of I2C receive function for START I2C.

HAL implementation of I2C receive function for ASF I2C.

HAL implementation of I2C receive function.



**Parameters**

in	<i>iface</i>	Device to interact with.
in	<i>word_address</i>	device transaction type
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**Parameters**

in	<i>iface</i>	Device to interact with.
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**Parameters**

in	<i>iface</i>	Device to interact with.
in	<i>word_address</i>	device word address
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.18 hal\_i2c\_release()**

```
ATCA_STATUS hal_i2c_release (
    void * hal_data )
```

manages reference count on given bus and releases resource if no more refences exist

manages reference count on given bus and releases resource if no more refernces exist

**Parameters**

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation
----	-----------------	---

## 18.9 Hardware abstraction layer (hal\_)

---

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### Parameters

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation return ATCA_SUCCESS
in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation

### Returns

ATCA\_SUCCESS

### 18.9.4.19 hal\_i2c\_send()

```
ATCA_STATUS hal_i2c_send (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * txdata,
    int txlength )
```

HAL implementation of I2C send over START.

HAL implementation of I2C send over ASF.

HAL implementation of I2C send.

### Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	device transaction type
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>txlength</i>	number of bytes to send

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### Parameters

in	<i>iface</i>	instance
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>txlength</i>	number of bytes to send

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**Parameters**

in	<i>iface</i>	instance
in	<i>word_address</i>	device word address
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>txlength</i>	number of bytes to send

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.20 hal\_i2c\_sleep()**

```
ATCA_STATUS hal_i2c_sleep (
    ATCAIface iface )
```

sleep CryptoAuth device using I2C bus

**Parameters**

in	<i>iface</i>	interface to logical device to sleep
----	--------------	--------------------------------------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.21 hal\_i2c\_wake()**

```
ATCA_STATUS hal_i2c_wake (
    ATCAIface iface )
```

wake up CryptoAuth device using I2C bus

**Parameters**

in	<i>iface</i>	interface to logical device to wakeup
----	--------------	---------------------------------------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

### 18.9.4.22 hal\_iface\_init()

```
ATCA_STATUS hal_iface_init (
    ATCAIfaceCfg * cfg,
    ATCAHAL_t ** hal )
```

Standard HAL API for ATCA to initialize a physical interface.

#### Parameters

in	<i>cfg</i>	pointer to <a href="#">ATCAIfaceCfg</a> object
in	<i>hal</i>	pointer to <a href="#">ATCAHAL_t</a> intermediate data structure

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.9.4.23 hal\_iface\_register\_hal()

```
ATCA_STATUS hal_iface_register_hal (
    ATCAIfaceType iface_type,
    ATCAHAL_t * hal,
    ATCAHAL_t ** old )
```

Register/Replace a HAL with a.

#### Parameters

in	<i>iface_type</i>	- the type of physical interface to register
in	<i>hal</i>	pointer to the new <a href="#">ATCAHAL_t</a> structure to register
out	<i>old</i>	pointer to the existing <a href="#">ATCAHAL_t</a> structure

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.9.4.24 hal\_iface\_release()

```
ATCA_STATUS hal_iface_release (
    ATCAIfaceType iface_type,
    void * hal_data )
```

releases a physical interface, HAL knows how to interpret hal\_data

**Parameters**

in	<i>iface_type</i>	- the type of physical interface to release
in	<i>hal_data</i>	- pointer to opaque hal data maintained by HAL implementation for this interface type

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.25 hal\_kit\_hid\_discover\_buses()**

```
ATCA_STATUS hal_kit_hid_discover_buses (
    int hid_buses[],
    int max_buses )
```

discover cdc buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge

discover all HID kits available. This function is currently not implemented. this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge

discover hid buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge. This function is currently not implemented.

**Parameters**

in	<i>hid_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover
in	<i>hid_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover

**Returns**

ATCA\_UNIMPLEMENTED

**Parameters**

in	<i>hid_buses</i>	an array of logical bus numbers
in	<i>max_buses</i>	maximum number of buses the app wants to attempt to discover

**Returns**

ATCA\_UNIMPLEMENTED

### 18.9.4.26 hal\_kit\_hid\_discover\_devices()

```
ATCA_STATUS hal_kit_hid_discover_devices (
    int bus_num,
    ATCAIfaceCfg cfg[],
    int * found )
```

discover any CryptoAuth devices on a given logical bus number

discover any CryptoAuth devices on a given logical bus number. This function is currently not implemented.

#### Parameters

in	<i>bus_num</i>	- logical bus number on which to look for CryptoAuth devices
out	<i>cfg[]</i>	- pointer to head of an array of interface config structures which get filled in by this method
out	<i>*found</i>	- number of devices found on this bus
in	<i>bus_num</i>	- logical bus number on which to look for CryptoAuth devices
out	<i>cfg[]</i>	- pointer to head of an array of interface config structures which get filled in by this method
out	<i>*found</i>	- number of devices found on this bus

#### Returns

ATCA\_UNIMPLEMENTED

### 18.9.4.27 hal\_kit\_hid\_idle()

```
ATCA_STATUS hal_kit_hid_idle (
    ATCAIface iface )
```

Call the idle for kit protocol.

Call the idle for kit protocol over USB HID.

#### Parameters

in	<i>iface</i>	ATCAIface instance that is the interface object to send the bytes over
----	--------------	--

#### Returns

ATCA\_STATUS

#### Parameters

in	<i>iface</i>	ATCAIface instance that is the interface object to send the bytes over
----	--------------	--

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.28 hal\_kit\_hid\_init()**

```
ATCA_STATUS hal_kit_hid_init (
    void * hal,
    ATCAIfaceCfg * cfg )
```

HAL implementation of Kit USB HID init.

**Parameters**

in	<i>hal</i>	pointer to HAL specific data that is maintained by this HAL
in	<i>cfg</i>	pointer to HAL specific configuration data that is used to initialize this HAL

**Returns**

ATCA\_STATUS

**Parameters**

in	<i>hal</i>	pointer to HAL specific data that is maintained by this HAL
in	<i>cfg</i>	pointer to HAL specific configuration data that is used to initialize this HAL

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.29 hal\_kit\_hid\_post\_init()**

```
ATCA_STATUS hal_kit_hid_post_init (
    ATCAIface iface )
```

HAL implementation of Kit HID post init.

**Parameters**

in	<i>iface</i>	instance
----	--------------	----------

**Returns**

ATCA\_STATUS

## 18.9 Hardware abstraction layer (hal\_)

---

### Parameters

in	<i>iface</i>	instance
----	--------------	----------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.9.4.30 hal\_kit\_hid\_receive()

```
ATCA_STATUS hal_kit_hid_receive (
    ATCAiface iface,
    uint8_t word_address,
    uint8_t * rxdata,
    uint16_t * rxsize )
```

HAL implementation of send over USB HID.

### Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	determine device transaction type
in	<i>rxdata</i>	pointer to space to receive the data
in, out	<i>rxsize</i>	ptr to expected number of receive bytes to request

### Returns

ATCA\_STATUS

### Parameters

in	<i>iface</i>	instance
in	<i>rxdata</i>	pointer to space to receive the data
in, out	<i>rxsize</i>	ptr to expected number of receive bytes to request

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### Parameters

in	<i>iface</i>	instance
in	<i>rxdata</i>	pointer to space to receive the data
in, out	<i>rxsize</i>	ptr to expected number of receive bytes to request



**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.31 hal\_kit\_hid\_release()**

```
ATCA_STATUS hal_kit_hid_release (
    void * hal_data )
```

Close the physical port for HID.

**Parameters**

in	<i>hal_data</i>	The hardware abstraction data specific to this HAL
----	-----------------	--

**Returns**

ATCA\_STATUS

**Parameters**

in	<i>hal_data</i>	The hardware abstraction data specific to this HAL
----	-----------------	--

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.32 hal\_kit\_hid\_send()**

```
ATCA_STATUS hal_kit_hid_send (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * txdata,
    int txlength )
```

HAL implementation of kit protocol send over USB HID.

**Parameters**

in	<i>iface</i>	instance
in	<i>word_address</i>	determine device transaction type
in	<i>txdata</i>	pointer to bytes to send
in	<i>txlength</i>	number of bytes to send

### Returns

ATCA\_STATUS

### Parameters

in	<i>iface</i>	instance
in	<i>txdata</i>	pointer to bytes to send
in	<i>txlength</i>	number of bytes to send

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.9.4.33 hal\_kit\_hid\_sleep()

```
ATCA_STATUS hal_kit_hid_sleep (
    ATCAIface iface )
```

Call the sleep for kit protocol.

Call the sleep for kit protocol over USB HID.

### Parameters

in	<i>iface</i>	ATCAIface instance that is the interface object to send the bytes over
----	--------------	--

### Returns

ATCA\_STATUS

### Parameters

in	<i>iface</i>	ATCAIface instance that is the interface object to send the bytes over
----	--------------	--

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.9.4.34 hal\_kit\_hid\_wake()

```
ATCA_STATUS hal_kit_hid_wake (
    ATCAIface iface )
```

Call the wake for kit protocol.

Call the wake for kit protocol over USB HID.

**Parameters**

in	<i>iface</i>	ATCAIface instance that is the interface object to send the bytes over
----	--------------	--

**Returns**

ATCA\_STATUS

**Parameters**

in	<i>iface</i>	ATCAIface instance that is the interface object to send the bytes over
----	--------------	--

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.35 hal\_lock\_mutex()**

```
ATCA_STATUS hal_lock_mutex (
    void * pMutex )
```

**18.9.4.36 hal\_malloc()**

```
void* hal_malloc (
    size_t size )
```

**18.9.4.37 hal\_rtos\_delay\_ms()**

```
void hal_rtos_delay_ms (
    uint32_t delay )
```

Timer API implemented at the HAL level.

This function delays for a number of milliseconds.

You can override this function if you like to do something else in your system while delaying.

**Parameters**

in	<i>delay</i>	Number of milliseconds to delay
----	--------------	---------------------------------

### 18.9.4.38 hal\_spi\_discover\_buses()

```
ATCA_STATUS hal_spi_discover_buses (
    int spi_buses[],
    int max_buses )
```

discover spi buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge

#### Parameters

in	<i>spi_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover

#### Returns

ATCA\_SUCCESS

### 18.9.4.39 hal\_spi\_discover\_devices()

```
ATCA_STATUS hal_spi_discover_devices (
    int bus_num,
    ATCAIfaceCfg cfg[],
    int * found )
```

discover any TA100 devices on a given logical bus number

#### Parameters

in	<i>bus_num</i>	logical bus number on which to look for TA100 devices
out	<i>cfg</i>	pointer to head of an array of interface config structures which get filled in by this method
out	<i>found</i>	number of devices found on this bus

#### Returns

ATCA\_SUCCESS

### 18.9.4.40 hal\_spi\_idle()

```
ATCA_STATUS hal_spi_idle (
    ATCAIface iface )
```

idle TA100 device using SPI bus

**Parameters**

in	<i>iface</i>	interface to logical device to idle
----	--------------	-------------------------------------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.41 hal\_spi\_init()**

```
ATCA_STATUS hal_spi_init (
    void * hal,
    ATCAIfaceCfg * cfg )
```

initialize an SPI interface using given config

**Parameters**

in	<i>hal</i>	- opaque ptr to HAL data
in	<i>cfg</i>	- interface configuration

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.42 hal\_spi\_post\_init()**

```
ATCA_STATUS hal_spi_post_init (
    ATCAIface iface )
```

HAL implementation of SPI post init.

**Parameters**

in	<i>iface</i>	instance
----	--------------	----------

**Returns**

ATCA\_SUCCESS

### 18.9.4.43 hal\_spi\_receive()

```
ATCA_STATUS hal_spi_receive (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * rxdata,
    uint16_t * rxlength )
```

HAL implementation of SPI receive function for HARMONY SPI.

#### Parameters

in	<i>iface</i>	Device to interact with.
in	<i>word_address</i>	device transaction type
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.9.4.44 hal\_spi\_release()

```
ATCA_STATUS hal_spi_release (
    void * hal_data )
```

manages reference count on given bus and releases resource if no more references exist

#### Parameters

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation
----	-----------------	---

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.9.4.45 hal\_spi\_send()

```
ATCA_STATUS hal_spi_send (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * txdata,
    int txlength )
```

HAL implementation of SPI send over Harmony.

**Parameters**

in	<i>iface</i>	instance
in	<i>word_address</i>	device transaction type
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>txlength</i>	number of bytes to send

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.46 hal\_spi\_sleep()**

```
ATCA_STATUS hal_spi_sleep (
    ATCAIface iface )
```

sleep TA100 device using SPI bus

**Parameters**

in	<i>iface</i>	interface to logical device to sleep
----	--------------	--------------------------------------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.47 hal\_spi\_wake()**

```
ATCA_STATUS hal_spi_wake (
    ATCAIface iface )
```

wake up TA100 device using SPI bus

**Parameters**

in	<i>iface</i>	interface to logical device to wakeup
----	--------------	---------------------------------------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

### 18.9.4.48 hal\_swi\_discover\_buses()

```
ATCA_STATUS hal_swi_discover_buses (
    int swi_buses[],
    int max_buses )
```

discover swi buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge

#### Parameters

in	<i>swi_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover

#### Returns

ATCA\_SUCCESS

### 18.9.4.49 hal\_swi\_discover\_devices()

```
ATCA_STATUS hal_swi_discover_devices (
    int bus_num,
    ATCAIfaceCfg cfg[],
    int * found )
```

discover any CryptoAuth devices on a given logical bus number

#### Parameters

in	<i>bus_num</i>	- logical bus number on which to look for CryptoAuth devices
out	<i>cfg[]</i>	- pointer to head of an array of interface config structures which get filled in by this method
out	<i>*found</i>	- number of devices found on this bus

#### Returns

ATCA\_SUCCESS

default configuration, to be reused during discovery process

### 18.9.4.50 hal\_swi\_idle()

```
ATCA_STATUS hal_swi_idle (
    ATCAIface iface )
```

idle CryptoAuth device using SWI interface



**Parameters**

in	<i>iface</i>	interface to logical device to idle
----	--------------	-------------------------------------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.51 hal\_swi\_init()**

```
ATCA_STATUS hal_swi_init (
    void * hal,
    ATCAIfCfg * cfg )
```

hal\_swi\_init manages requests to initialize a physical interface. It manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIfCfg instances using the same bus, and you can have multiple ATCAIfCfg instances on multiple swi buses, so hal\_swi\_init manages these things and ATCAIfCfg is abstracted from the physical details.

initialize an SWI interface using given config

**Parameters**

in	<i>hal</i>	- opaque ptr to HAL data
in	<i>cfg</i>	- interface configuration

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.52 hal\_swi\_post\_init()**

```
ATCA_STATUS hal_swi_post_init (
    ATCAIfCfg iface )
```

HAL implementation of SWI post init.

**Parameters**

in	<i>iface</i>	instance
----	--------------	----------

**Returns**

ATCA\_SUCCESS

### 18.9.4.53 hal\_swi\_receive()

```
ATCA_STATUS hal_swi_receive (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * rxdata,
    uint16_t * rxlength )
```

HAL implementation of SWI receive function over UART.

#### Parameters

in	<i>iface</i>	Device to interact with.
in	<i>word_address</i>	device transaction type
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.9.4.54 hal\_swi\_release()

```
ATCA_STATUS hal_swi_release (
    void * hal_data )
```

manages reference count on given bus and releases resource if no more references exist

#### Parameters

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation
----	-----------------	---

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.9.4.55 hal\_swi\_send()

```
ATCA_STATUS hal_swi_send (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * txdata,
    int txlength )
```

HAL implementation of SWI send command over UART.

**Parameters**

in	<i>iface</i>	instance
in	<i>word_address</i>	device transaction type
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>txlength</i>	number of bytes to send

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.56 hal\_swi\_send\_flag()**

```
ATCA_STATUS hal_swi_send_flag (
    ATCAIface iface,
    uint8_t data )
```

HAL implementation of SWI send one byte over UART.

**Parameters**

in	<i>iface</i>	instance
in	<i>data</i>	bytes to send

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.57 hal\_swi\_sleep()**

```
ATCA_STATUS hal_swi_sleep (
    ATCAIface iface )
```

sleep CryptoAuth device using SWI interface

**Parameters**

in	<i>iface</i>	interface to logical device to sleep
----	--------------	--------------------------------------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

### 18.9.4.58 hal\_swi\_wake()

```
ATCA_STATUS hal_swi_wake (
    ATCAIface iface )
```

wake up CryptoAuth device using SWI interface

#### Parameters

in	<i>iface</i>	interface to logical device to wakeup
----	--------------	---------------------------------------

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.9.4.59 hal\_unlock\_mutex()

```
ATCA_STATUS hal_unlock_mutex (
    void * pMutex )
```

### 18.9.4.60 kit\_id\_from\_devtype()

```
const char* kit_id_from_devtype (
    ATCADeviceType devtype )
```

Kit Protocol is key

### 18.9.4.61 kit\_idle()

```
ATCA_STATUS kit_idle (
    ATCAIface iface )
```

Call the idle for kit protocol.

#### Parameters

in	<i>iface</i>	the interface object to send the bytes over
----	--------------	---

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.62 kit\_init()**

```
ATCA_STATUS kit_init (
    ATCAIface iface )
```

HAL implementation of kit protocol init. This function calls back to the physical protocol to send the bytes.

**Parameters**

in	<i>iface</i>	instance
----	--------------	----------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.63 kit\_interface\_from\_kittype()**

```
const char* kit_interface_from_kittype (
    ATCAKitType kittype )
```

Kit interface from device

**18.9.4.64 kit\_parse\_rsp()**

```
ATCA_STATUS kit_parse_rsp (
    const char * pkitbuf,
    int nkitbuf,
    uint8_t * kitstatus,
    uint8_t * rxdata,
    int * datasize )
```

Parse the response ascii from the kit.

**Parameters**

out	<i>pkitbuf</i>	pointer to ascii kit protocol data to parse
in	<i>nkitbuf</i>	length of the ascii kit protocol data
in	<i>kitstatus</i>	status of the ascii device
in	<i>rxdata</i>	pointer to the binary data buffer
in	<i>datasize</i>	size of the pointer to the binary data buffer

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

### 18.9.4.65 kit\_phy\_num\_found()

```
ATCA_STATUS kit_phy_num_found (
    int8_t * num_found )
```

Number of USB HID devices found.

#### Parameters

out	<i>num_found</i>	return number of USB HID devices found
-----	------------------	--

#### Returns

ATCA\_STATUS

#### Parameters

out	<i>num_found</i>	
-----	------------------	--

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### Parameters

out	<i>num_found</i>	
-----	------------------	--

#### Returns

SUCCESS

### 18.9.4.66 kit\_phy\_receive() [1/2]

```
ATCA_STATUS kit_phy_receive (
    ATCAIface iface,
    char * rxdata,
    int * rxsize )
```

HAL implementation of kit protocol receive data. It is called by the top layer.

#### Parameters

in	<i>iface</i>	instance
out	<i>rxdata</i>	pointer to space to receive the data
in, out	<i>rxsize</i>	ptr to expected number of receive bytes to request

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.67 kit\_phy\_receive() [2/2]**

```
ATCA_STATUS kit_phy_receive (
    ATCAIface iface,
    uint8_t * rxdata,
    int * rxsize )
```

HAL implementation of kit protocol send over USB HID.

HAL implementation of kit protocol receive. This function is called by the top layer.

**Parameters**

in	<i>iface</i>	instance
out	<i>rxdata</i>	pointer to space to receive the data
in, out	<i>rxsize</i>	ptr to expected number of receive bytes to request

**Returns**

ATCA\_STATUS

**Parameters**

in	<i>iface</i>	instance
out	<i>rxdata</i>	pointer to space to receive the data
in, out	<i>rxsize</i>	ptr to expected number of receive bytes to request

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.68 kit\_phy\_send() [1/2]**

```
ATCA_STATUS kit_phy_send (
    ATCAIface iface,
    const char * txdata,
    int txlength )
```

HAL implementation of kit protocol send .It is called by the top layer.

### Parameters

in	<i>iface</i>	instance
in	<i>txdata</i>	pointer to bytes to send
in	<i>txlength</i>	number of bytes to send

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.9.4.69 kit\_phy\_send() [2/2]

```
ATCA_STATUS kit_phy_send (  
    ATCAIface iface,  
    uint8_t * txdata,  
    int txlength )
```

HAL implementation of send over USB HID.

HAL implementation of send over Kit protocol. This function is called by the top layer.

### Parameters

in	<i>iface</i>	instance
in	<i>txdata</i>	pointer to bytes to send
in	<i>txlength</i>	number of bytes to send

### Returns

ATCA\_STATUS

### Parameters

in	<i>iface</i>	instance
in	<i>txdata</i>	pointer to bytes to send
in	<i>txlength</i>	number of bytes to send

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.9.4.70 kit\_receive()

```
ATCA_STATUS kit_receive (  
    ATCAIface iface,
```



```
uint8_t word_address,
uint8_t * rxdata,
uint16_t * rxsize )
```

HAL implementation to receive bytes and unwrap from kit protocol. This function calls back to the physical protocol to receive the bytes.

#### Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	device transaction type
in	<i>rxdata</i>	pointer to space to receive the data
in, out	<i>rxsize</i>	ptr to expected number of receive bytes to request

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.9.4.71 kit\_send()

```
ATCA_STATUS kit_send (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * txdata,
    int txlength )
```

HAL implementation of kit protocol send. This function calls back to the physical protocol to send the bytes.

#### Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	device transaction type
in	<i>txdata</i>	pointer to bytes to send
in	<i>txlength</i>	number of bytes to send

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.9.4.72 kit\_sleep()

```
ATCA_STATUS kit_sleep (
    ATCAIface iface )
```

Call the sleep for kit protocol.

### Parameters

in	<i>iface</i>	the interface object to send the bytes over
----	--------------	---

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.9.4.73 kit\_wake()

```
ATCA_STATUS kit_wake (  
    ATCAIface iface )
```

Call the wake for kit protocol.

### Parameters

in	<i>iface</i>	the interface object to send the bytes over
----	--------------	---

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.9.4.74 kit\_wrap\_cmd()

```
ATCA_STATUS kit_wrap_cmd (  
    const uint8_t * txdata,  
    int txlen,  
    char * pkitcmd,  
    int * nkitcmd,  
    char target )
```

Wrap binary bytes in ascii kit protocol.

### Parameters

in	<i>txdata</i>	Binary data to wrap.
in	<i>txlen</i>	Length of binary data in bytes.
out	<i>pkitcmd</i>	ASCII kit protocol wrapped data is return here.
in, out	<i>nkitcmd</i>	As input, the size of the pkitcmd buffer. As output, the number of bytes returned in the pkitcmd buffer.
in	<i>target</i>	Target char to use 's' for SHA devices, 'e' for ECC devices.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.75 strnchr()**

```
char* strnchr (
    const char * s,
    size_t count,
    int c )
```

**18.9.4.76 swi\_uart\_deinit()**

```
ATCA_STATUS swi_uart_deinit (
    ATCASWIMaster_t * instance )
```

Implementation of SWI UART deinit.

HAL implementation of SWI UART deinit.

**Parameters**

in	<i>instance</i>	instance
----	-----------------	----------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**Parameters**

in	<i>instance</i>	instance
----	-----------------	----------

**Returns**

ATCA\_SUCCESS

**18.9.4.77 swi\_uart\_discover\_buses()**

```
void swi_uart_discover_buses (
    int swi_uart_buses[],
    int max_buses )
```

discover UART buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge

### Parameters

in	<i>swi_uart_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover

### 18.9.4.78 swi\_uart\_init()

```
ATCA_STATUS swi_uart_init (  
    ATCASWIMaster_t * instance )
```

Implementation of SWI UART init.

HAL implementation of SWI UART init.

- this HAL implementation assumes you've included the ASF SERCOM UART libraries in your project, otherwise, the HAL layer will not compile because the ASF UART drivers are a dependency \*

### Parameters

in	<i>instance</i>	instance
----	-----------------	----------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

- this HAL implementation assumes you've included the START SERCOM UART libraries in your project, otherwise, the HAL layer will not compile because the START UART drivers are a dependency \*

### Parameters

in	<i>instance</i>	instance
----	-----------------	----------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.9.4.79 swi\_uart\_mode()

```
void swi_uart_mode (  
    ATCASWIMaster_t * instance,  
    uint8_t mode )
```

implementation of SWI UART change mode.

HAL implementation of SWI UART change mode.

## Parameters

in	<i>instance</i>	instance
in	<i>mode</i>	(TRANSMIT_MODE or RECEIVE_MODE)

**18.9.4.80 swi\_uart\_receive\_byte()**

```
ATCA_STATUS swi_uart_receive_byte (
    ATCASWIMaster_t * instance,
    uint8_t * data )
```

HAL implementation of SWI UART receive bytes over ASF. This function receive one byte over UART.

## Parameters

in	<i>instance</i>	instance
out	<i>data</i>	pointer to space to receive the data

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.81 swi\_uart\_send\_byte()**

```
ATCA_STATUS swi_uart_send_byte (
    ATCASWIMaster_t * instance,
    uint8_t data )
```

HAL implementation of SWI UART send byte over ASF. This function send one byte over UART.

## Parameters

in	<i>instance</i>	instance
in	<i>data</i>	number of byte to send

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.9.4.82 swi\_uart\_setbaud()**

```
void swi_uart_setbaud (
    ATCASWIMaster_t * instance,
    uint32_t baudrate )
```

implementation of SWI UART change baudrate.

HAL implementation of SWI UART change baudrate.

### Parameters

in	<i>instance</i>	instance
in	<i>baudrate</i>	(typically 230400 , 160000 or 115200)
in	<i>instance</i>	instance
in	<i>baudrate</i>	(typically 230400 or 115200)

## 18.9.5 Variable Documentation

### 18.9.5.1 \_\_gHid [1/3]

```
atcahid_t __gHid
```

### 18.9.5.2 \_\_gHid [2/3]

```
atcahid_t __gHid
```

### 18.9.5.3 \_\_gHid [3/3]

```
atcahid_t __gHid
```

### 18.9.5.4 pin\_conf

```
struct port_config pin_conf
```

## 18.10 Host side crypto methods (atcah\_)

Use these functions if your system does not use an ATCADevice as a host but implements the host in firmware. The functions provide host-side cryptographic functionality for an ATECC client device. They are intended to accompany the CryptoAuthLib functions. They can be called directly from an application, or integrated into an API.

### Data Structures

- struct [atca\\_temp\\_key](#)  
*Structure to hold TempKey fields.*
- struct [atca\\_include\\_data\\_in\\_out](#)  
*Input / output parameters for function [atca\\_include\\_data\(\)](#).*
- struct [atca\\_nonce\\_in\\_out](#)  
*Input/output parameters for function [atca\\_nonce\(\)](#).*
- struct [atca\\_io\\_decrypt\\_in\\_out](#)
- struct [atca\\_verify\\_mac](#)
- struct [atca\\_secureboot\\_enc\\_in\\_out](#)
- struct [atca\\_secureboot\\_mac\\_in\\_out](#)
- struct [atca\\_mac\\_in\\_out](#)  
*Input/output parameters for function [atca\\_mac\(\)](#).*
- struct [atca\\_hmac\\_in\\_out](#)  
*Input/output parameters for function [atca\\_hmac\(\)](#).*
- struct [atca\\_gen\\_dig\\_in\\_out](#)  
*Input/output parameters for function [atcah\\_gen\\_dig\(\)](#).*
- struct [atca\\_write\\_mac\\_in\\_out](#)  
*Input/output parameters for function [atcah\\_write\\_auth\\_mac\(\)](#) and [atcah\\_privwrite\\_auth\\_mac\(\)](#).*
- struct [atca\\_derive\\_key\\_in\\_out](#)  
*Input/output parameters for function [atcah\\_derive\\_key\(\)](#).*
- struct [atca\\_derive\\_key\\_mac\\_in\\_out](#)  
*Input/output parameters for function [atcah\\_derive\\_key\\_mac\(\)](#).*
- struct [atca\\_decrypt\\_in\\_out](#)  
*Input/output parameters for function [atca\\_decrypt\(\)](#).*
- struct [atca\\_check\\_mac\\_in\\_out](#)  
*Input/output parameters for function [atcah\\_check\\_mac\(\)](#).*
- struct [atca\\_verify\\_in\\_out](#)  
*Input/output parameters for function [atcah\\_verify\(\)](#).*
- struct [atca\\_gen\\_key\\_in\\_out](#)  
*Input/output parameters for calculating the PubKey digest put into TempKey by the GenKey command with the [atcah\\_gen\\_key\\_msg\(\)](#) function.*
- struct [atca\\_sign\\_internal\\_in\\_out](#)  
*Input/output parameters for calculating the message and digest used by the Sign(internal) command. Used with the [atcah\\_sign\\_internal\\_msg\(\)](#) function.*

## Typedefs

- typedef struct [atca\\_temp\\_key](#) [atca\\_temp\\_key\\_t](#)  
*Structure to hold TempKey fields.*
- typedef struct [atca\\_nonce\\_in\\_out](#) [atca\\_nonce\\_in\\_out\\_t](#)
- typedef struct [atca\\_io\\_decrypt\\_in\\_out](#) [atca\\_io\\_decrypt\\_in\\_out\\_t](#)
- typedef struct [atca\\_verify\\_mac](#) [atca\\_verify\\_mac\\_in\\_out\\_t](#)
- typedef struct [atca\\_secureboot\\_enc\\_in\\_out](#) [atca\\_secureboot\\_enc\\_in\\_out\\_t](#)
- typedef struct [atca\\_secureboot\\_mac\\_in\\_out](#) [atca\\_secureboot\\_mac\\_in\\_out\\_t](#)
- typedef struct [atca\\_mac\\_in\\_out](#) [atca\\_mac\\_in\\_out\\_t](#)
- typedef struct [atca\\_gen\\_dig\\_in\\_out](#) [atca\\_gen\\_dig\\_in\\_out\\_t](#)  
*Input/output parameters for function [atcah\\_gen\\_dig\(\)](#).*
- typedef struct [atca\\_write\\_mac\\_in\\_out](#) [atca\\_write\\_mac\\_in\\_out\\_t](#)  
*Input/output parameters for function [atcah\\_write\\_auth\\_mac\(\)](#) and [atcah\\_privwrite\\_auth\\_mac\(\)](#).*
- typedef struct [atca\\_check\\_mac\\_in\\_out](#) [atca\\_check\\_mac\\_in\\_out\\_t](#)  
*Input/output parameters for function [atcah\\_check\\_mac\(\)](#).*
- typedef struct [atca\\_verify\\_in\\_out](#) [atca\\_verify\\_in\\_out\\_t](#)
- typedef struct [atca\\_gen\\_key\\_in\\_out](#) [atca\\_gen\\_key\\_in\\_out\\_t](#)  
*Input/output parameters for calculating the PubKey digest put into TempKey by the GenKey command with the [atcah\\_gen\\_key\\_msg\(\)](#) function.*
- typedef struct [atca\\_sign\\_internal\\_in\\_out](#) [atca\\_sign\\_internal\\_in\\_out\\_t](#)  
*Input/output parameters for calculating the message and digest used by the Sign(internal) command. Used with the [atcah\\_sign\\_internal\\_msg\(\)](#) function.*

## Functions

- [ATCA\\_STATUS atcah\\_nonce](#) (struct [atca\\_nonce\\_in\\_out](#) \*param)  
*This function calculates host side nonce with the parameters passed.*
- [ATCA\\_STATUS atcah\\_mac](#) (struct [atca\\_mac\\_in\\_out](#) \*param)  
*This function generates an SHA-256 digest (MAC) of a key, challenge, and other information.*
- [ATCA\\_STATUS atcah\\_check\\_mac](#) (struct [atca\\_check\\_mac\\_in\\_out](#) \*param)  
*This function performs the checkmac operation to generate client response on the host side .*
- [ATCA\\_STATUS atcah\\_hmac](#) (struct [atca\\_hmac\\_in\\_out](#) \*param)  
*This function generates an HMAC / SHA-256 hash of a key and other information.*
- [ATCA\\_STATUS atcah\\_gen\\_dig](#) (struct [atca\\_gen\\_dig\\_in\\_out](#) \*param)  
*This function combines the current TempKey with a stored value.*
- [ATCA\\_STATUS atcah\\_gen\\_mac](#) (struct [atca\\_gen\\_dig\\_in\\_out](#) \*param)  
*This function generates mac with session key with a plain text.*
- [ATCA\\_STATUS atcah\\_write\\_auth\\_mac](#) (struct [atca\\_write\\_mac\\_in\\_out](#) \*param)  
*This function calculates the input MAC for the Write command.*
- [ATCA\\_STATUS atcah\\_privwrite\\_auth\\_mac](#) (struct [atca\\_write\\_mac\\_in\\_out](#) \*param)  
*This function calculates the input MAC for the PrivWrite command.*
- [ATCA\\_STATUS atcah\\_derive\\_key](#) (struct [atca\\_derive\\_key\\_in\\_out](#) \*param)  
*This function derives a key with a key and TempKey.*
- [ATCA\\_STATUS atcah\\_derive\\_key\\_mac](#) (struct [atca\\_derive\\_key\\_mac\\_in\\_out](#) \*param)  
*This function calculates the input MAC for a DeriveKey command.*
- [ATCA\\_STATUS atcah\\_decrypt](#) (struct [atca\\_decrypt\\_in\\_out](#) \*param)  
*This function decrypts 32-byte encrypted data received with the Read command.*
- [ATCA\\_STATUS atcah\\_sha256](#) (int32\_t len, const uint8\_t \*message, uint8\_t \*digest)  
*This function creates a SHA256 digest on a little-endian system.*
- uint8\_t \* [atcah\\_include\\_data](#) (struct [atca\\_include\\_data\\_in\\_out](#) \*param)



- This function copies otp and sn data into a command buffer.*
- **ATCA\_STATUS atcah\_gen\_key\_msg** (struct **atca\_gen\_key\_in\_out** \*param)  
*Calculate the PubKey digest created by GenKey and saved to TempKey.*
  - **ATCA\_STATUS atcah\_config\_to\_sign\_internal** (ATCADeviceType device\_type, struct **atca\_sign\_internal\_in\_out** \*param, const uint8\_t \*config)  
*Populate the slot\_config, key\_config, and is\_slot\_locked fields in the atca\_sign\_internal\_in\_out structure from the provided config zone.*
  - **ATCA\_STATUS atcah\_sign\_internal\_msg** (ATCADeviceType device\_type, struct **atca\_sign\_internal\_in\_out** \*param)  
*Builds the full message that would be signed by the Sign(Internal) command.*
  - **ATCA\_STATUS atcah\_verify\_mac** (atca\_verify\_mac\_in\_out\_t \*param)  
*Calculate the expected MAC on the host side for the Verify command.*
  - **ATCA\_STATUS atcah\_secureboot\_enc** (atca\_secureboot\_enc\_in\_out\_t \*param)  
*Encrypts the digest for the SecureBoot command when using the encrypted digest / validating mac option.*
  - **ATCA\_STATUS atcah\_secureboot\_mac** (atca\_secureboot\_mac\_in\_out\_t \*param)  
*Calculates the expected MAC returned from the SecureBoot command when verification is a success.*
  - **ATCA\_STATUS atcah\_encode\_counter\_match** (uint32\_t counter, uint8\_t \*counter\_match)  
*Builds the counter match value that needs to be stored in a slot.*
  - **ATCA\_STATUS atcah\_io\_decrypt** (struct **atca\_io\_decrypt\_in\_out** \*param)  
*Decrypt data that's been encrypted by the IO protection key. The ECDH and KDF commands on the ATECC608 are the only ones that support this operation.*

## Variables

- uint8\_t \* **p\_temp**  
*[out] pointer to output buffer*
- const uint8\_t \* **otp**  
*[in] pointer to one-time-programming data*
- const uint8\_t \* **sn**  
*[in] pointer to serial number data*
- uint8\_t **mode**  
*[in] Mode parameter used in Nonce command (Param1).*
- uint16\_t **zero**  
*[in] Zero parameter used in Nonce command (Param2).*
- const uint8\_t \* **num\_in**  
*[in] Pointer to 20-byte NumIn data used in Nonce command.*
- const uint8\_t \* **rand\_out**  
*[in] Pointer to 32-byte RandOut data from Nonce command.*
- struct **atca\_temp\_key** \* **temp\_key**  
*[in,out] Pointer to TempKey structure.*
- uint8\_t **mode**  
*[in] Mode parameter used in MAC command (Param1).*
- uint16\_t **key\_id**  
*[in] KeyID parameter used in MAC command (Param2).*
- const uint8\_t \* **challenge**  
*[in] Pointer to 32-byte Challenge data used in MAC command, depending on mode.*
- const uint8\_t \* **key**  
*[in] Pointer to 32-byte key used to generate MAC digest.*
- const uint8\_t \* **otp**  
*[in] Pointer to 11-byte OTP, optionally included in MAC digest, depending on mode.*
- const uint8\_t \* **sn**

- [in]* Pointer to 9-byte SN, optionally included in MAC digest, depending on mode.
- uint8\_t \* [response](#)
  - [out]* Pointer to 32-byte SHA-256 digest (MAC).
- struct [atca\\_temp\\_key](#) \* [temp\\_key](#)
  - [in,out]* Pointer to TempKey structure.
- uint8\_t [mode](#)
  - [in]* Mode parameter used in HMAC command (Param1).
- uint16\_t [key\\_id](#)
  - [in]* KeyID parameter used in HMAC command (Param2).
- const uint8\_t \* [key](#)
  - [in]* Pointer to 32-byte key used to generate HMAC digest.
- const uint8\_t \* [otp](#)
  - [in]* Pointer to 11-byte OTP, optionally included in HMAC digest, depending on mode.
- const uint8\_t \* [sn](#)
  - [in]* Pointer to 9-byte SN, optionally included in HMAC digest, depending on mode.
- uint8\_t \* [response](#)
  - [out]* Pointer to 32-byte SHA-256 HMAC digest.
- struct [atca\\_temp\\_key](#) \* [temp\\_key](#)
  - [in,out]* Pointer to TempKey structure.
- uint8\_t \* [crypto\\_data](#)
  - [in,out]* Pointer to 32-byte data. Input encrypted data from Read command (Contents field), output decrypted.
- struct [atca\\_temp\\_key](#) \* [temp\\_key](#)
  - [in,out]* Pointer to TempKey structure.
- uint16\_t [curve\\_type](#)
  - [in]* Curve type used in Verify command (Param2).
- const uint8\_t \* [signature](#)
  - [in]* Pointer to ECDSA signature to be verified
- const uint8\_t \* [public\\_key](#)
  - [in]* Pointer to the public key to be used for verification
- struct [atca\\_temp\\_key](#) \* [temp\\_key](#)
  - [in,out]* Pointer to TempKey structure.

## Definitions for ATECC Message Sizes to Calculate a SHA256 Hash

"||" is the concatenation operator. The number in braces is the length of the hash input value in bytes.

- #define [ATCA\\_MSG\\_SIZE\\_NONCE](#) (55)
  - RandOut{32} || NumIn{20} || OpCode{1} || Mode{1} || LSB of Param2{1}.*
- #define [ATCA\\_MSG\\_SIZE\\_MAC](#) (88)
  - (Key or TempKey){32} || (Challenge or TempKey){32} || OpCode{1} || Mode{1} || Param2{2} || (OTP0\_7 or 0){8} || (OTP8\_10 or 0){3} || SN8{1} || (SN4\_7 or 0){4} || SN0\_1{2} || (SN2\_3 or 0){2}*
- #define [ATCA\\_MSG\\_SIZE\\_HMAC](#) (88)
- #define [ATCA\\_MSG\\_SIZE\\_GEN\\_DIG](#) (96)
  - KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0\_1{2} || 0{25} || TempKey{32}.*
- #define [ATCA\\_MSG\\_SIZE\\_DERIVE\\_KEY](#) (96)
  - KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0\_1{2} || 0{25} || TempKey{32}.*
- #define [ATCA\\_MSG\\_SIZE\\_DERIVE\\_KEY\\_MAC](#) (39)
  - KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0\_1{2}.*
- #define [ATCA\\_MSG\\_SIZE\\_ENCRYPT\\_MAC](#) (96)

- KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0\_1{2} || 0{25} || TempKey{32}.*
- #define `ATCA_MSG_SIZE_PRIVWRITE_MAC` (96)
  - KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0\_1{2} || 0{21} || PlainText{36}.*
- #define `ATCA_COMMAND_HEADER_SIZE` ( 4)
- #define `ATCA_GENDIG_ZEROS_SIZE` (25)
- #define `ATCA_WRITE_MAC_ZEROS_SIZE` (25)
- #define `ATCA_PRIVWRITE_MAC_ZEROS_SIZE` (21)
- #define `ATCA_PRIVWRITE_PLAIN_TEXT_SIZE` (36)
- #define `ATCA_DERIVE_KEY_ZEROS_SIZE` (25)
- #define `ATCA_HMAC_BLOCK_SIZE` (64)
- #define `ENCRYPTION_KEY_SIZE` (64)

## Default Fixed Byte Values of Serial Number (SN[0:1] and SN[8])

- #define `ATCA_SN_0_DEF` (0x01)
- #define `ATCA_SN_1_DEF` (0x23)
- #define `ATCA_SN_8_DEF` (0xEE)

## Definition for TempKey Mode

- #define `MAC_MODE_USE_TEMPKEY_MASK` ((uint8\_t)0x03)  
*mode mask for MAC command when using TempKey*

### 18.10.1 Detailed Description

Use these functions if your system does not use an ATCADevice as a host but implements the host in firmware. The functions provide host-side cryptographic functionality for an ATECC client device. They are intended to accompany the CryptoAuthLib functions. They can be called directly from an application, or integrated into an API.

Modern compilers can garbage-collect unused functions. If your compiler does not support this feature, you can just discard this module from your project if you do use an ATECC as a host. Or, if you don't, delete the functions you do not use.

### 18.10.2 Macro Definition Documentation

#### 18.10.2.1 ATCA\_COMMAND\_HEADER\_SIZE

```
#define ATCA_COMMAND_HEADER_SIZE ( 4)
```

#### 18.10.2.2 ATCA\_DERIVE\_KEY\_ZEROS\_SIZE

```
#define ATCA_DERIVE_KEY_ZEROS_SIZE (25)
```

### 18.10.2.3 ATCA\_GENDIG\_ZEROS\_SIZE

```
#define ATCA_GENDIG_ZEROS_SIZE (25)
```

### 18.10.2.4 ATCA\_HMAC\_BLOCK\_SIZE

```
#define ATCA_HMAC_BLOCK_SIZE (64)
```

### 18.10.2.5 ATCA\_MSG\_SIZE\_DERIVE\_KEY

```
#define ATCA_MSG_SIZE_DERIVE_KEY (96)
```

```
KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2} || 0{25} || TempKey{32}.
```

### 18.10.2.6 ATCA\_MSG\_SIZE\_DERIVE\_KEY\_MAC

```
#define ATCA_MSG_SIZE_DERIVE_KEY_MAC (39)
```

```
KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2}.
```

### 18.10.2.7 ATCA\_MSG\_SIZE\_ENCRYPT\_MAC

```
#define ATCA_MSG_SIZE_ENCRYPT_MAC (96)
```

```
KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2} || 0{25} || TempKey{32}.
```

### 18.10.2.8 ATCA\_MSG\_SIZE\_GEN\_DIG

```
#define ATCA_MSG_SIZE_GEN_DIG (96)
```

```
KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0_1{2} || 0{25} || TempKey{32}.
```

### 18.10.2.9 ATCA\_MSG\_SIZE\_HMAC

```
#define ATCA_MSG_SIZE_HMAC (88)
```

**18.10.2.10 ATCA\_MSG\_SIZE\_MAC**

```
#define ATCA_MSG_SIZE_MAC (88)
```

(Key or TempKey){32} || (Challenge or TempKey){32} || OpCode{1} || Mode{1} || Param2{2} || (OTP0\_7 or 0){8} || (OTP8\_10 or 0){3} || SN8{1} || (SN4\_7 or 0){4} || SN0\_1{2} || (SN2\_3 or 0){2}

**18.10.2.11 ATCA\_MSG\_SIZE\_NONCE**

```
#define ATCA_MSG_SIZE_NONCE (55)
```

RandOut{32} || NumIn{20} || OpCode{1} || Mode{1} || LSB of Param2{1}.

**18.10.2.12 ATCA\_MSG\_SIZE\_PRIVWRITE\_MAC**

```
#define ATCA_MSG_SIZE_PRIVWRITE_MAC (96)
```

KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0\_1{2} || 0{21} || PlainText{36}.

**18.10.2.13 ATCA\_PRIVWRITE\_MAC\_ZEROS\_SIZE**

```
#define ATCA_PRIVWRITE_MAC_ZEROS_SIZE (21)
```

**18.10.2.14 ATCA\_PRIVWRITE\_PLAIN\_TEXT\_SIZE**

```
#define ATCA_PRIVWRITE_PLAIN_TEXT_SIZE (36)
```

**18.10.2.15 ATCA\_SN\_0\_DEF**

```
#define ATCA_SN_0_DEF (0x01)
```

**18.10.2.16 ATCA\_SN\_1\_DEF**

```
#define ATCA_SN_1_DEF (0x23)
```

### 18.10.2.17 ATCA\_SN\_8\_DEF

```
#define ATCA_SN_8_DEF (0xEE)
```

### 18.10.2.18 ATCA\_WRITE\_MAC\_ZEROS\_SIZE

```
#define ATCA_WRITE_MAC_ZEROS_SIZE (25)
```

### 18.10.2.19 ENCRYPTION\_KEY\_SIZE

```
#define ENCRYPTION_KEY_SIZE (64)
```

### 18.10.2.20 MAC\_MODE\_USE\_TEMPKEY\_MASK

```
#define MAC_MODE_USE_TEMPKEY_MASK ((uint8_t) 0x03)
```

mode mask for MAC command when using TempKey

## 18.10.3 Typedef Documentation

### 18.10.3.1 atca\_check\_mac\_in\_out\_t

```
typedef struct atca_check_mac_in_out atca_check_mac_in_out_t
```

Input/output parameters for function [atcah\\_check\\_mac\(\)](#).

### 18.10.3.2 atca\_gen\_dig\_in\_out\_t

```
typedef struct atca_gen_dig_in_out atca_gen_dig_in_out_t
```

Input/output parameters for function [atcah\\_gen\\_dig\(\)](#).

### 18.10.3.3 atca\_gen\_key\_in\_out\_t

```
typedef struct atca_gen_key_in_out atca_gen_key_in_out_t
```

Input/output parameters for calculating the PubKey digest put into TempKey by the GenKey command with the [atcah\\_gen\\_key\\_msg\(\)](#) function.

### 18.10.3.4 atca\_io\_decrypt\_in\_out\_t

```
typedef struct atca_io_decrypt_in_out atca_io_decrypt_in_out_t
```

### 18.10.3.5 atca\_mac\_in\_out\_t

```
typedef struct atca_mac_in_out atca_mac_in_out_t
```

### 18.10.3.6 atca\_nonce\_in\_out\_t

```
typedef struct atca_nonce_in_out atca_nonce_in_out_t
```

### 18.10.3.7 atca\_secureboot\_enc\_in\_out\_t

```
typedef struct atca_secureboot_enc_in_out atca_secureboot_enc_in_out_t
```

### 18.10.3.8 atca\_secureboot\_mac\_in\_out\_t

```
typedef struct atca_secureboot_mac_in_out atca_secureboot_mac_in_out_t
```

### 18.10.3.9 atca\_sign\_internal\_in\_out\_t

```
typedef struct atca_sign_internal_in_out atca_sign_internal_in_out_t
```

Input/output parameters for calculating the message and digest used by the Sign(internal) command. Used with the [atcah\\_sign\\_internal\\_msg\(\)](#) function.

### 18.10.3.10 atca\_temp\_key\_t

```
typedef struct atca_temp_key atca_temp_key_t
```

Structure to hold TempKey fields.

### 18.10.3.11 atca\_verify\_in\_out\_t

```
typedef struct atca_verify_in_out atca_verify_in_out_t
```

### 18.10.3.12 atca\_verify\_mac\_in\_out\_t

```
typedef struct atca_verify_mac atca_verify_mac_in_out_t
```

### 18.10.3.13 atca\_write\_mac\_in\_out\_t

```
typedef struct atca_write_mac_in_out atca_write_mac_in_out_t
```

Input/output parameters for function [atcah\\_write\\_auth\\_mac\(\)](#) and [atcah\\_privwrite\\_auth\\_mac\(\)](#).

## 18.10.4 Function Documentation

### 18.10.4.1 atcah\_check\_mac()

```
ATCA_STATUS atcah_check_mac (
    struct atca_check_mac_in_out * param )
```

This function performs the checkmac operation to generate client response on the host side .

#### Parameters

in, out	param	Input and output parameters
---------	-------	-----------------------------

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.



### 18.10.4.2 atcah\_config\_to\_sign\_internal()

```
ATCA_STATUS atcah_config_to_sign_internal (
    ATCADeviceType device_type,
    struct atca_sign_internal_in_out * param,
    const uint8_t * config )
```

Populate the slot\_config, key\_config, and is\_slot\_locked fields in the [atca\\_sign\\_internal\\_in\\_out](#) structure from the provided config zone.

The [atca\\_sign\\_internal\\_in\\_out](#) structure has a number of fields (slot\_config, key\_config, is\_slot\_locked) that can be determined automatically from the current state of TempKey and the full config zone.

#### Parameters

in, out	<i>param</i>	Sign(Internal) parameters to be filled out. Only slot_config, key_config, and is_slot_locked will be set.
in	<i>device_type</i>	The type of the device.
in	<i>config</i>	Full 128 byte config zone for the device.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.10.4.3 atcah\_decrypt()

```
ATCA_STATUS atcah_decrypt (
    struct atca_decrypt_in_out * param )
```

This function decrypts 32-byte encrypted data received with the Read command.

To use this function, first the nonce must be valid and synchronized between device and application. The application sends a GenDig command to the Device, using a key specified by SlotConfig.ReadKey. The device updates its TempKey. The application then updates its own TempKey using the GenDig calculation function, using the same key. The application sends a Read command to the device for a user zone configured with EncryptRead. The device encrypts 32-byte zone content, and outputs it to the host. The application passes these encrypted data to this decryption function. The function decrypts the data and returns them. TempKey must be updated by GenDig using a ParentKey as specified by SlotConfig.ReadKey before executing this function. The decryption function does not check whether the TempKey has been generated by a correct ParentKey for the corresponding zone. Therefore to get a correct result, the application has to make sure that prior GenDig calculation was done using correct ParentKey.

#### Parameters

in, out	<i>param</i>	pointer to parameter structure
---------	--------------	--------------------------------

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.10.4.4 atcah\_derive\_key()

```
ATCA_STATUS atcah_derive_key (
    struct atca_derive_key_in_out * param )
```

This function derives a key with a key and TempKey.

Used in conjunction with DeriveKey command, the key derived by this function will match the key in the device. Two kinds of operation are supported:

- Roll Key operation: target\_key and parent\_key parameters should be set to point to the same location (TargetKey).
- Create Key operation: target\_key should be set to point to TargetKey, parent\_key should be set to point to ParentKey.

After executing this function, the initial value of target\_key will be overwritten with the derived key. The TempKey should be valid (temp\_key.valid = 1) before executing this function.

#### Parameters

in, out	param	pointer to parameter structure
---------	-------	--------------------------------

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.10.4.5 atcah\_derive\_key\_mac()

```
ATCA_STATUS atcah_derive_key_mac (
    struct atca_derive_key_mac_in_out * param )
```

This function calculates the input MAC for a DeriveKey command.

The DeriveKey command will need an input MAC if SlotConfig[TargetKey].Bit15 is set.

#### Parameters

in, out	param	pointer to parameter structure
---------	-------	--------------------------------

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.10.4.6 atcah\_encode\_counter\_match()

```
ATCA_STATUS atcah_encode_counter_match (
    uint32_t counter_value,
    uint8_t * counter_match_value )
```

Builds the counter match value that needs to be stored in a slot.

##### Parameters

in	<i>counter_value</i>	Counter value to be used for the counter match. This must be a multiple of 32.
out	<i>counter_match_value</i>	Data to be stored in the beginning of a counter match slot will be returned here (8 bytes).

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.10.4.7 atcah\_gen\_dig()

```
ATCA_STATUS atcah_gen_dig (
    struct atca_gen_dig_in_out * param )
```

This function combines the current TempKey with a stored value.

The stored value can be a data slot, OTP page, configuration zone, or hardware transport key. The TempKey generated by this function will match with the TempKey in the device generated when executing a GenDig command. The TempKey should be valid (`temp_key.valid = 1`) before executing this function. To use this function, an application first sends a GenDig command with a chosen stored value to the device. This stored value must be known by the application and is passed to this GenDig calculation function. The function calculates a new TempKey and returns it.

##### Parameters

in, out	<i>param</i>	pointer to parameter structure
---------	--------------	--------------------------------

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.10.4.8 atcah\_gen\_key\_msg()

```
ATCA_STATUS atcah_gen_key_msg (
    struct atca_gen_key_in_out * param )
```

Calculate the PubKey digest created by GenKey and saved to TempKey.

## 18.10 Host side crypto methods (atcah\_)

---

### Parameters

<i>in, out</i>	<i>param</i>	GenKey parameters required to calculate the PubKey digest. Digest is return in the temp_key parameter.
----------------	--------------	--

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.10.4.9 atcah\_gen\_mac()

```
ATCA_STATUS atcah_gen_mac (
    struct atca_gen_dig_in_out * param )
```

This function generates mac with session key with a plain text.

### Parameters

<i>in, out</i>	<i>param</i>	pointer to parameter structure
----------------	--------------	--------------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.10.4.10 atcah\_hmac()

```
ATCA_STATUS atcah_hmac (
    struct atca_hmac_in_out * param )
```

This function generates an HMAC / SHA-256 hash of a key and other information.

The resulting hash will match with the one generated in the device by an HMAC command. The TempKey has to be valid (temp\_key.valid = 1) before executing this function.

### Parameters

<i>in, out</i>	<i>param</i>	pointer to parameter structure
----------------	--------------	--------------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

**18.10.4.11 atcah\_include\_data()**

```
uint8_t* atcah_include_data (
    struct atca_include_data_in_out * param )
```

This function copies otp and sn data into a command buffer.

**Parameters**

in, out	<i>param</i>	pointer to parameter structure
---------	--------------	--------------------------------

**Returns**

pointer to command buffer byte that was copied last

**18.10.4.12 atcah\_io\_decrypt()**

```
ATCA_STATUS atcah_io_decrypt (
    struct atca_io_decrypt_in_out * param )
```

Decrypt data that's been encrypted by the IO protection key. The ECDH and KDF commands on the ATECC608 are the only ones that support this operation.

**Parameters**

in, out	<i>param</i>	Parameters required to perform the operation.
---------	--------------	---

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.10.4.13 atcah\_mac()**

```
ATCA_STATUS atcah_mac (
    struct atca_mac_in_out * param )
```

This function generates an SHA-256 digest (MAC) of a key, challenge, and other information.

The resulting digest will match with the one generated by the device when executing a MAC command. The Temp↔Key (if used) should be valid (temp\_key.valid = 1) before executing this function.

**Parameters**

in, out	<i>param</i>	pointer to parameter structure
---------	--------------	--------------------------------

## 18.10 Host side crypto methods (atcah\_)

---

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.10.4.14 atcah\_nonce()

```
ATCA_STATUS atcah_nonce (
    struct atca_nonce_in_out * param )
```

This function calculates host side nonce with the parameters passed.

### Parameters

in, out	param	pointer to parameter structure
---------	-------	--------------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.10.4.15 atcah\_privwrite\_auth\_mac()

```
ATCA_STATUS atcah_privwrite_auth_mac (
    struct atca_write_mac_in_out * param )
```

This function calculates the input MAC for the PrivWrite command.

The PrivWrite command will need an input MAC if SlotConfig.WriteConfig.Encrypt is set.

### Parameters

in, out	param	pointer to parameter structure
---------	-------	--------------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 18.10.4.16 atcah\_secureboot\_enc()

```
ATCA_STATUS atcah_secureboot_enc (
    atca_secureboot_enc_in_out_t * param )
```

Encrypts the digest for the SecureBoot command when using the encrypted digest / validating mac option.

**Parameters**

<i>in, out</i>	<i>param</i>	Data required to perform the operation.
----------------	--------------	---

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.10.4.17 atcah\_secureboot\_mac()**

```
ATCA_STATUS atcah_secureboot_mac (
    atca_secureboot_mac_in_out_t * param )
```

Calculates the expected MAC returned from the SecureBoot command when verification is a success.

The result of this function (param->mac) should be compared with the actual MAC returned to validate the response.

**Parameters**

<i>in, out</i>	<i>param</i>	Data required to perform the operation.
----------------	--------------	---

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.10.4.18 atcah\_sha256()**

```
ATCA_STATUS atcah_sha256 (
    int32_t len,
    const uint8_t * message,
    uint8_t * digest )
```

This function creates a SHA256 digest on a little-endian system.

**Parameters**

<i>in</i>	<i>len</i>	byte length of message
<i>in</i>	<i>message</i>	pointer to message
<i>out</i>	<i>digest</i>	SHA256 of message

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

### 18.10.4.19 atcah\_sign\_internal\_msg()

```
ATCA_STATUS atcah_sign_internal_msg (
    ATCADeviceType device_type,
    struct atca_sign_internal_in_out * param )
```

Builds the full message that would be signed by the Sign(Internal) command.

Additionally, the function will optionally output the OtherData data required by the Verify(In/Validate) command as well as the SHA256 digest of the full message.

#### Parameters

out	<i>device_type</i>	Device type to perform the calculation for.
out	<i>param</i>	Input data and output buffers required.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.10.4.20 atcah\_verify\_mac()

```
ATCA_STATUS atcah_verify_mac (
    atca_verify_mac_in_out_t * param )
```

Calculate the expected MAC on the host side for the Verify command.

#### Parameters

in, out	<i>param</i>	Data required to perform the operation.
---------	--------------	---

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.10.4.21 atcah\_write\_auth\_mac()

```
ATCA_STATUS atcah_write_auth_mac (
    struct atca_write_mac_in_out * param )
```

This function calculates the input MAC for the Write command.

The Write command will need an input MAC if SlotConfig.WriteConfig.Encrypt is set.



**Parameters**

<code>in, out</code>	<code>param</code>	pointer to parameter structure
----------------------	--------------------	--------------------------------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

## 18.10.5 Variable Documentation

### 18.10.5.1 challenge

`challenge`

[in] Pointer to 32-byte Challenge data used in MAC command, depending on mode.

### 18.10.5.2 crypto\_data

`crypto_data`

[in,out] Pointer to 32-byte data. Input encrypted data from Read command (Contents field), output decrypted.

### 18.10.5.3 curve\_type

`curve_type`

[in] Curve type used in Verify command (Param2).

### 18.10.5.4 key [1/2]

`key`

[in] Pointer to 32-byte key used to generate MAC digest.

### 18.10.5.5 key [2/2]

key

[in] Pointer to 32-byte key used to generate HMAC digest.

### 18.10.5.6 key\_id [1/2]

key\_id

[in] KeyID parameter used in MAC command (Param2).

### 18.10.5.7 key\_id [2/2]

key\_id

[in] KeyID parameter used in HMAC command (Param2).

### 18.10.5.8 mode [1/3]

mode

[in] Mode parameter used in Nonce command (Param1).

### 18.10.5.9 mode [2/3]

mode

[in] Mode parameter used in MAC command (Param1).

### 18.10.5.10 mode [3/3]

mode

[in] Mode parameter used in HMAC command (Param1).

**18.10.5.11 num\_in**

num\_in

[in] Pointer to 20-byte NumIn data used in Nonce command.

**18.10.5.12 otp [1/3]**

otp

[in] pointer to one-time-programming data

**18.10.5.13 otp [2/3]**

otp

[in] Pointer to 11-byte OTP, optionally included in MAC digest, depending on mode.

**18.10.5.14 otp [3/3]**

otp

[in] Pointer to 11-byte OTP, optionally included in HMAC digest, depending on mode.

**18.10.5.15 p\_temp**

p\_temp

[out] pointer to output buffer

**18.10.5.16 public\_key**

public\_key

[in] Pointer to the public key to be used for verification

### 18.10.5.17 rand\_out

rand\_out

[in] Pointer to 32-byte RandOut data from Nonce command.

### 18.10.5.18 response [1/2]

response

[out] Pointer to 32-byte SHA-256 digest (MAC).

### 18.10.5.19 response [2/2]

response

[out] Pointer to 32-byte SHA-256 HMAC digest.

### 18.10.5.20 signature

signature

[in] Pointer to ECDSA signature to be verified

### 18.10.5.21 sn [1/3]

sn

[in] pointer to serial number data

### 18.10.5.22 sn [2/3]

sn

[in] Pointer to 9-byte SN, optionally included in MAC digest, depending on mode.

**18.10.5.23 sn [3/3]**

sn

[in] Pointer to 9-byte SN, optionally included in HMAC digest, depending on mode.

**18.10.5.24 temp\_key [1/5]**

temp\_key

[in,out] Pointer to TempKey structure.

**18.10.5.25 temp\_key [2/5]**

temp\_key

[in,out] Pointer to TempKey structure.

**18.10.5.26 temp\_key [3/5]**

temp\_key

[in,out] Pointer to TempKey structure.

**18.10.5.27 temp\_key [4/5]**

temp\_key

[in,out] Pointer to TempKey structure.

**18.10.5.28 temp\_key [5/5]**

temp\_key

[in,out] Pointer to TempKey structure.

**18.10.5.29 zero**

zero

[in] Zero parameter used in Nonce command (Param2).

## 18.11 JSON Web Token (JWT) methods (atca\_jwt\_)

Methods for signing and verifying JSON Web Token (JWT) tokens.

### Data Structures

- struct [atca\\_jwt\\_t](#)  
*Structure to hold metadata information about the jwt being built.*

### Functions

- [ATCA\\_STATUS atca\\_jwt\\_init](#) ([atca\\_jwt\\_t](#) \*jwt, char \*buf, uint16\_t buflen)  
*Initialize a JWT structure.*
- [ATCA\\_STATUS atca\\_jwt\\_add\\_claim\\_string](#) ([atca\\_jwt\\_t](#) \*jwt, const char \*claim, const char \*value)  
*Add a string claim to a token.*
- [ATCA\\_STATUS atca\\_jwt\\_add\\_claim\\_numeric](#) ([atca\\_jwt\\_t](#) \*jwt, const char \*claim, int32\_t value)  
*Add a numeric claim to a token.*
- [ATCA\\_STATUS atca\\_jwt\\_finalize](#) ([atca\\_jwt\\_t](#) \*jwt, uint16\_t key\_id)  
*Close the claims of a token, encode them, then sign the result.*
- void [atca\\_jwt\\_check\\_payload\\_start](#) ([atca\\_jwt\\_t](#) \*jwt)  
*Check the provided context to see what character needs to be added in order to append a claim.*
- [ATCA\\_STATUS atca\\_jwt\\_verify](#) (const char \*buf, uint16\_t buflen, const uint8\_t \*pubkey)  
*Verifies the signature of a jwt using the provided public key.*

### 18.11.1 Detailed Description

Methods for signing and verifying JSON Web Token (JWT) tokens.

### 18.11.2 Function Documentation

#### 18.11.2.1 atca\_jwt\_add\_claim\_numeric()

```
ATCA_STATUS atca_jwt_add_claim_numeric (
    atca_jwt_t * jwt,
    const char * claim,
    int32_t value )
```

Add a numeric claim to a token.

#### Note

This function does not escape strings so the user has to ensure the claim is valid first

### 18.11.2.2 atca\_jwt\_add\_claim\_string()

```
ATCA_STATUS atca_jwt_add_claim_string (
    atca_jwt_t * jwt,
    const char * claim,
    const char * value )
```

Add a string claim to a token.

#### Note

This function does not escape strings so the user has to ensure they are valid for use in a JSON string first

### 18.11.2.3 atca\_jwt\_check\_payload\_start()

```
void atca_jwt_check_payload_start (
    atca_jwt_t * jwt )
```

Check the provided context to see what character needs to be added in order to append a claim.

### 18.11.2.4 atca\_jwt\_finalize()

```
ATCA_STATUS atca_jwt_finalize (
    atca_jwt_t * jwt,
    uint16_t key_id )
```

Close the claims of a token, encode them, then sign the result.

### 18.11.2.5 atca\_jwt\_init()

```
ATCA_STATUS atca_jwt_init (
    atca_jwt_t * jwt,
    char * buf,
    uint16_t buflen )
```

Initialize a JWT structure.

### 18.11.2.6 atca\_jwt\_verify()

```
ATCA_STATUS atca_jwt_verify (
    const char * buf,
    uint16_t buflen,
    const uint8_t * pubkey )
```

Verifies the signature of a jwt using the provided public key.

## 18.12 mbedTLS Wrapper methods (atca\_mbedtls\_)

These methods are for interfacing cryptoauthlib to mbedtls.

### Functions

- int [atca\\_mbedtls\\_pk\\_init](#) (struct mbedtls\_pk\_context \*pkey, const uint16\_t slotid)  
*Initializes an mbedtls pk context for use with EC operations.*
- int [atca\\_mbedtls\\_cert\\_add](#) (struct mbedtls\_x509\_crt \*cert, const struct [atcacert\\_def\\_s](#) \*cert\_def)
- int [atca\\_mbedtls\\_ecdh\\_slot\\_cb](#) (void)  
*ECDH Callback to obtain the "slot" used in ECDH operations from the application.*
- int [atca\\_mbedtls\\_ecdh\\_ioprot\\_cb](#) (uint8\_t secret[32])  
*ECDH Callback to obtain the IO Protection secret from the application.*

### 18.12.1 Detailed Description

These methods are for interfacing cryptoauthlib to mbedtls.

### 18.12.2 Function Documentation

#### 18.12.2.1 atca\_mbedtls\_cert\_add()

```
int atca_mbedtls_cert_add (
    struct mbedtls_x509_crt * cert,
    const struct atcacert\_def\_s * cert_def )
```

#### 18.12.2.2 atca\_mbedtls\_ecdh\_ioprot\_cb()

```
int atca_mbedtls_ecdh_ioprot_cb (
    uint8_t secret[32] )
```

ECDH Callback to obtain the IO Protection secret from the application.

#### Parameters

out	<i>secret</i>	32 byte array used to store the secret
-----	---------------	--

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.



18.12.2.3 atca\_mbedtls\_ecdh\_slot\_cb()

```
int atca_mbedtls_ecdh_slot_cb (  
    void )
```

ECDH Callback to obtain the "slot" used in ECDH operations from the application.

Returns

Slot Number

18.12.2.4 atca\_mbedtls\_pk\_init()

```
int atca_mbedtls_pk_init (  
    mbedtls_pk_context * pkey,  
    const uint16_t slotid )
```

Initializes an mbedtls pk context for use with EC operations.

Parameters

in, out	<i>pkey</i>	ptr to space to receive version string
in	<i>slotid</i>	Associated with this key

Returns

0 on success, otherwise an error code.

## 18.13 Attributes (pkcs11\_attrib\_)

### Data Structures

- struct [\\_pkcs11\\_mech\\_table\\_e](#)

### Macros

- `#define` [PKCS11\\_MECH\\_ECC508\\_EC\\_CAPABILITY](#) ([CKF\\_EC\\_F\\_P](#) | [CKF\\_EC\\_NAMEDCURVE](#) | [CKF\\_EC\\_UNCOMPRESS](#))
- `#define` [TABLE\\_SIZE\(x\)](#) `sizeof(x) / sizeof(x[0])`

### Typedefs

- `typedef struct` [\\_pkcs11\\_mech\\_table\\_e](#) [pkcs11\\_mech\\_table\\_e](#)
- `typedef struct` [\\_pkcs11\\_mech\\_table\\_e](#) \* [pkcs11\\_mech\\_table\\_ptr](#)

### Functions

- [CK\\_RV](#) [pkcs11\\_attrib\\_fill](#) ([CK\\_ATTRIBUTE\\_PTR](#) pAttribute, const [CK\\_VOID\\_PTR](#) pData, const [CK\\_ULONG](#) ulSize)  
*Perform the necessary checks and copy data into an attribute structure.*
- [CK\\_RV](#) [pkcs11\\_attrib\\_value](#) ([CK\\_ATTRIBUTE\\_PTR](#) pAttribute, const [CK\\_ULONG](#) ulValue, const [CK\\_ULONG](#) ulSize)  
*Helper function to write a numerical value to an attribute buffer.*
- [CK\\_RV](#) [pkcs11\\_attrib\\_false](#) (const [CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV](#) [pkcs11\\_attrib\\_true](#) (const [CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV](#) [pkcs11\\_attrib\\_empty](#) (const [CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV](#) [pkcs11\\_cert\\_get\\_encoded](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV](#) [pkcs11\\_cert\\_get\\_type](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV](#) [pkcs11\\_cert\\_get\\_subject](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV](#) [pkcs11\\_cert\\_get\\_subject\\_key\\_id](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV](#) [pkcs11\\_cert\\_get\\_authority\\_key\\_id](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV](#) [pkcs11\\_cert\\_get\\_trusted\\_flag](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV](#) [pkcs11\\_cert\\_x509\\_write](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- void [pkcs11\\_config\\_init\\_private](#) ([pkcs11\\_object\\_ptr](#) pObject, char \*label, size\_t len)
- void [pkcs11\\_config\\_init\\_public](#) ([pkcs11\\_object\\_ptr](#) pObject, char \*label, size\_t len)
- void [pkcs11\\_config\\_init\\_cert](#) ([pkcs11\\_object\\_ptr](#) pObject, char \*label, size\_t len)
- [CK\\_RV](#) [pkcs11\\_config\\_cert](#) ([pkcs11\\_lib\\_ctx\\_ptr](#) pLibCtx, [pkcs11\\_slot\\_ctx\\_ptr](#) pSlot, [pkcs11\\_object\\_ptr](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pLabel)
- [CK\\_RV](#) [pkcs11\\_config\\_key](#) ([pkcs11\\_lib\\_ctx\\_ptr](#) pLibCtx, [pkcs11\\_slot\\_ctx\\_ptr](#) pSlot, [pkcs11\\_object\\_ptr](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pLabel)
- [CK\\_RV](#) [pkcs11\\_config\\_remove\\_object](#) ([pkcs11\\_lib\\_ctx\\_ptr](#) pLibCtx, [pkcs11\\_slot\\_ctx\\_ptr](#) pSlot, [pkcs11\\_object\\_ptr](#) pObject)
- [CK\\_RV](#) [pkcs11\\_config\\_load\\_objects](#) ([pkcs11\\_slot\\_ctx\\_ptr](#) slot\_ctx)
- [CK\\_RV](#) [pkcs11\\_config\\_load](#) ([pkcs11\\_slot\\_ctx\\_ptr](#) slot\_ctx)
- [CK\\_RV](#) [pkcs11\\_find\\_init](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount)
- [CK\\_RV](#) [pkcs11\\_find\\_continue](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phObject, [CK\\_ULONG](#) ulMaxObjectCount, [CK\\_ULONG\\_PTR](#) pulObjectCount)
- [CK\\_RV](#) [pkcs11\\_find\\_finish](#) ([CK\\_SESSION\\_HANDLE](#) hSession)

- [CK\\_RV pkcs11\\_find\\_get\\_attribute](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE](#) hObject, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount)
- [CK\\_RV pkcs11\\_get\\_lib\\_info](#) ([CK\\_INFO\\_PTR](#) pInfo)  
*Obtains general information about Cryptoki.*
- [pkcs11\\_lib\\_ctx\\_ptr pkcs11\\_get\\_context](#) (void)  
*Retrieve the current library context.*
- [CK\\_RV pkcs11\\_lock\\_context](#) ([pkcs11\\_lib\\_ctx\\_ptr](#) pContext)
- [CK\\_RV pkcs11\\_unlock\\_context](#) ([pkcs11\\_lib\\_ctx\\_ptr](#) pContext)
- [CK\\_RV pkcs11\\_init\\_check](#) ([pkcs11\\_lib\\_ctx\\_ptr](#) \*ppContext, [CK\\_BBOOL](#) lock)  
*Check if the library is initialized properly.*
- [CK\\_RV pkcs11\\_init](#) ([CK\\_C\\_INITIALIZE\\_ARGS\\_PTR](#) pInitArgs)  
*Initializes the PKCS11 API Library for Cryptoauthlib.*
- [CK\\_RV pkcs11\\_deinit](#) ([CK\\_VOID\\_PTR](#) pReserved)
- [CK\\_RV pkcs11\\_key\\_write](#) ([CK\\_VOID\\_PTR](#) pSession, [CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_key\\_generate\\_pair](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_ATTRIBUTE\\_PTR](#) pPublicKeyTemplate, [CK\\_ULONG](#) ulPublicKeyAttributeCount, [CK\\_ATTRIBUTE\\_PTR](#) pPrivateKeyTemplate, [CK\\_ULONG](#) ulPrivateKeyAttributeCount, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phPublicKey, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phPrivateKey)
- [CK\\_RV pkcs11\\_key\\_derive](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_OBJECT\\_HANDLE](#) hBaseKey, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phKey)
- [CK\\_RV C\\_Initialize](#) ([CK\\_VOID\\_PTR](#) pInitArgs)  
*Initializes Cryptoki library NOTES: If pInitArgs is a non-NULL\_PTR is must dereference to a [CK\\_C\\_INITIALIZE\\_ARGS](#) structure.*
- [CK\\_RV C\\_Finalize](#) ([CK\\_VOID\\_PTR](#) pReserved)  
*Clean up miscellaneous Cryptoki-associated resources.*
- [CK\\_RV C\\_GetInfo](#) ([CK\\_INFO\\_PTR](#) pInfo)  
*Obtains general information about Cryptoki.*
- [CK\\_RV C\\_GetFunctionList](#) ([CK\\_FUNCTION\\_LIST\\_PTR\\_PTR](#) ppFunctionList)  
*Obtains entry points of Cryptoki library functions.*
- [CK\\_RV C\\_GetSlotList](#) ([CK\\_BBOOL](#) tokenPresent, [CK\\_SLOT\\_ID\\_PTR](#) pSlotList, [CK\\_ULONG\\_PTR](#) pulCount)  
*Obtains a list of slots in the system.*
- [CK\\_RV C\\_GetSlotInfo](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_SLOT\\_INFO\\_PTR](#) pInfo)  
*Obtains information about a particular slot.*
- [CK\\_RV C\\_GetTokenInfo](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_TOKEN\\_INFO\\_PTR](#) pInfo)  
*Obtains information about a particular token.*
- [CK\\_RV C\\_GetMechanismList](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_MECHANISM\\_TYPE\\_PTR](#) pMechanismList, [CK\\_ULONG\\_PTR](#) pulCount)  
*Obtains a list of mechanisms supported by a token (in a slot)*
- [CK\\_RV C\\_GetMechanismInfo](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_MECHANISM\\_TYPE](#) type, [CK\\_MECHANISM\\_INFO\\_PTR](#) pInfo)  
*Obtains information about a particular mechanism of a token (in a slot)*
- [CK\\_RV C\\_InitToken](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_UTF8CHAR\\_PTR](#) pPin, [CK\\_ULONG](#) ulPinLen, [CK\\_UTF8CHAR\\_PTR](#) pLabel)  
*Initializes a token (in a slot)*
- [CK\\_RV C\\_InitPIN](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_UTF8CHAR\\_PTR](#) pPin, [CK\\_ULONG](#) ulPinLen)  
*Initializes the normal user's PIN.*
- [CK\\_RV C\\_SetPIN](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_UTF8CHAR\\_PTR](#) pOldPin, [CK\\_ULONG](#) ulOldLen, [CK\\_UTF8CHAR\\_PTR](#) pNewPin, [CK\\_ULONG](#) ulNewLen)  
*Modifies the PIN of the current user.*

- [CK\\_RV C\\_OpenSession](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_FLAGS](#) flags, [CK\\_VOID\\_PTR](#) pApplication, [CK\\_NOTIFY](#) notify, [CK\\_SESSION\\_HANDLE\\_PTR](#) phSession)  
*Opens a connection between an application and a particular token or sets up an application callback for token insertion.*
- [CK\\_RV C\\_CloseSession](#) ([CK\\_SESSION\\_HANDLE](#) hSession)  
*Close the given session.*
- [CK\\_RV C\\_CloseAllSessions](#) ([CK\\_SLOT\\_ID](#) slotID)  
*Close all open sessions.*
- [CK\\_RV C\\_GetSessionInfo](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_SESSION\\_INFO\\_PTR](#) pInfo)  
*Retrieve information about the specified session.*
- [CK\\_RV C\\_GetOperationState](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pOperationState, [CK\\_ULONG\\_PTR](#) pulOperationStateLen)  
*Obtains the cryptographic operations state of a session.*
- [CK\\_RV C\\_SetOperationState](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pOperationState, [CK\\_ULONG](#) ulOperationStateLen, [CK\\_OBJECT\\_HANDLE](#) hEncryptionKey, [CK\\_OBJECT\\_HANDLE](#) hAuthenticationKey)  
*Sets the cryptographic operations state of a session.*
- [CK\\_RV C\\_Login](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_USER\\_TYPE](#) userType, [CK\\_UTF8CHAR\\_PTR](#) pPin, [CK\\_ULONG](#) ulPinLen)  
*Login on the token in the specified session.*
- [CK\\_RV C\\_Logout](#) ([CK\\_SESSION\\_HANDLE](#) hSession)  
*Log out of the token in the specified session.*
- [CK\\_RV C\\_CreateObject](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phObject)  
*Create a new object on the token in the specified session using the given attribute template.*
- [CK\\_RV C\\_CopyObject](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE](#) hObject, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phNewObject)  
*Create a copy of the object with the specified handle.*
- [CK\\_RV C\\_DestroyObject](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE](#) hObject)  
*Destroy the specified object.*
- [CK\\_RV C\\_GetObjectSize](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE](#) hObject, [CK\\_ULONG\\_PTR](#) pulSize)  
*Obtains the size of an object in bytes.*
- [CK\\_RV C\\_GetAttributeValue](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE](#) hObject, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount)  
*Obtains an attribute value of an object.*
- [CK\\_RV C\\_SetAttributeValue](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE](#) hObject, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount)  
*Change or set the value of the specified attributes on the specified object.*
- [CK\\_RV C\\_FindObjectsInit](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount)  
*Initializes an object search in the specified session using the specified attribute template as search parameters.*
- [CK\\_RV C\\_FindObjects](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phObject, [CK\\_ULONG](#) ulMaxObjectCount, [CK\\_ULONG\\_PTR](#) pulObjectCount)  
*Continue the search for objects in the specified session.*
- [CK\\_RV C\\_FindObjectsFinal](#) ([CK\\_SESSION\\_HANDLE](#) hSession)  
*Finishes an object search operation (and cleans up)*
- [CK\\_RV C\\_EncryptInit](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_OBJECT\\_HANDLE](#) hObject)  
*Initializes an encryption operation using the specified mechanism and session.*
- [CK\\_RV C\\_Encrypt](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG](#) ulDataLen, [CK\\_BYTE\\_PTR](#) pEncryptedData, [CK\\_ULONG\\_PTR](#) pulEncryptedDataLen)  
*Perform a single operation encryption operation in the specified session.*

- [CK\\_RV C\\_EncryptUpdate](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG](#) ulDataLen, [CK\\_BYTE\\_PTR](#) pEncryptedData, [CK\\_ULONG\\_PTR](#) pulEncryptedDataLen)  
*Continues a multiple-part encryption operation.*
- [CK\\_RV C\\_EncryptFinal](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pEncryptedData, [CK\\_ULONG\\_PTR](#) pulEncryptedDataLen)  
*Finishes a multiple-part encryption operation.*
- [CK\\_RV C\\_DecryptInit](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_OBJECT\\_HANDLE](#) hObject)  
*Initialize decryption using the specified object.*
- [CK\\_RV C\\_Decrypt](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pEncryptedData, [CK\\_ULONG](#) ulEncryptedDataLen, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG\\_PTR](#) pulDataLen)  
*Perform a single operation decryption in the given session.*
- [CK\\_RV C\\_DecryptUpdate](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pEncryptedData, [CK\\_ULONG](#) ulEncryptedDataLen, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG\\_PTR](#) pulDataLen)  
*Continues a multiple-part decryption operation.*
- [CK\\_RV C\\_DecryptFinal](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG\\_PTR](#) pulDataLen)  
*Finishes a multiple-part decryption operation.*
- [CK\\_RV C\\_DigestInit](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism)  
*Initializes a message-digesting operation using the specified mechanism in the specified session.*
- [CK\\_RV C\\_Digest](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG](#) ulDataLen, [CK\\_BYTE\\_PTR](#) pDigest, [CK\\_ULONG\\_PTR](#) pulDigestLen)  
*Digest the specified data in a one-pass operation and return the resulting digest.*
- [CK\\_RV C\\_DigestUpdate](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pPart, [CK\\_ULONG](#) ulPartLen)  
*Continues a multiple-part digesting operation.*
- [CK\\_RV C\\_DigestKey](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE](#) hObject)  
*Update a running digest operation by digesting a secret key with the specified handle.*
- [CK\\_RV C\\_DigestFinal](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pDigest, [CK\\_ULONG\\_PTR](#) pulDigestLen)  
*Finishes a multiple-part digesting operation.*
- [CK\\_RV C\\_SignInit](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_OBJECT\\_HANDLE](#) hKey)  
*Initialize a signing operation using the specified key and mechanism.*
- [CK\\_RV C\\_Sign](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG](#) ulDataLen, [CK\\_BYTE\\_PTR](#) pSignature, [CK\\_ULONG\\_PTR](#) pulSignatureLen)  
*Sign the data in a single pass operation.*
- [CK\\_RV C\\_SignUpdate](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pPart, [CK\\_ULONG](#) ulPartLen)  
*Continues a multiple-part signature operation.*
- [CK\\_RV C\\_SignFinal](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pSignature, [CK\\_ULONG\\_PTR](#) pulSignatureLen)  
*Finishes a multiple-part signature operation.*
- [CK\\_RV C\\_SignRecoverInit](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_OBJECT\\_HANDLE](#) hKey)  
*Initializes a signature operation, where the data can be recovered from the signature.*
- [CK\\_RV C\\_SignRecover](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG](#) ulDataLen, [CK\\_BYTE\\_PTR](#) pSignature, [CK\\_ULONG\\_PTR](#) pulSignatureLen)  
*Signs single-part data, where the data can be recovered from the signature.*
- [CK\\_RV C\\_VerifyInit](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_OBJECT\\_HANDLE](#) hKey)  
*Initializes a verification operation using the specified key and mechanism.*
- [CK\\_RV C\\_Verify](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG](#) ulDataLen, [CK\\_BYTE\\_PTR](#) pSignature, [CK\\_ULONG](#) ulSignatureLen)  
*Verifies a signature on single-part data.*

- **CK\_RV C\_VerifyUpdate** (CK\_SESSION\_HANDLE hSession, CK\_BYTE\_PTR pPart, CK\_ULONG ulPartLen)  
*Continues a multiple-part verification operation.*
- **CK\_RV C\_VerifyFinal** (CK\_SESSION\_HANDLE hSession, CK\_BYTE\_PTR pSignature, CK\_ULONG ulSignatureLen)  
*Finishes a multiple-part verification operation.*
- **CK\_RV C\_VerifyRecoverInit** (CK\_SESSION\_HANDLE hSession, CK\_MECHANISM\_PTR pMechanism, CK\_OBJECT\_HANDLE hKey)  
*Initializes a verification operation where the data is recovered from the signature.*
- **CK\_RV C\_VerifyRecover** (CK\_SESSION\_HANDLE hSession, CK\_BYTE\_PTR pSignature, CK\_ULONG ulSignatureLen, CK\_BYTE\_PTR pData, CK\_ULONG\_PTR pulDataLen)  
*Verifies a signature on single-part data, where the data is recovered from the signature.*
- **CK\_RV C\_DigestEncryptUpdate** (CK\_SESSION\_HANDLE hSession, CK\_BYTE\_PTR pPart, CK\_ULONG ulPartLen, CK\_BYTE\_PTR pEncryptedPart, CK\_ULONG\_PTR pulEncryptedPartLen)  
*Continues simultaneous multiple-part digesting and encryption operations.*
- **CK\_RV C\_DecryptDigestUpdate** (CK\_SESSION\_HANDLE hSession, CK\_BYTE\_PTR pPart, CK\_ULONG ulPartLen, CK\_BYTE\_PTR pDecryptedPart, CK\_ULONG\_PTR pulDecryptedPartLen)  
*Continues simultaneous multiple-part decryption and digesting operations.*
- **CK\_RV C\_SignEncryptUpdate** (CK\_SESSION\_HANDLE hSession, CK\_BYTE\_PTR pPart, CK\_ULONG ulPartLen, CK\_BYTE\_PTR pEncryptedPart, CK\_ULONG\_PTR pulEncryptedPartLen)  
*Continues simultaneous multiple-part signature and encryption operations.*
- **CK\_RV C\_DecryptVerifyUpdate** (CK\_SESSION\_HANDLE hSession, CK\_BYTE\_PTR pEncryptedPart, CK\_ULONG ulEncryptedPartLen, CK\_BYTE\_PTR pPart, CK\_ULONG\_PTR pulPartLen)  
*Continues simultaneous multiple-part decryption and verification operations.*
- **CK\_RV C\_GenerateKey** (CK\_SESSION\_HANDLE hSession, CK\_MECHANISM\_PTR pMechanism, CK\_ATTRIBUTE\_PTR pTemplate, CK\_ULONG ulCount, CK\_OBJECT\_HANDLE\_PTR phKey)  
*Generates a secret key using the specified mechanism.*
- **CK\_RV C\_GenerateKeyPair** (CK\_SESSION\_HANDLE hSession, CK\_MECHANISM\_PTR pMechanism, CK\_ATTRIBUTE\_PTR pPublicKeyTemplate, CK\_ULONG ulPublicKeyAttributeCount, CK\_ATTRIBUTE\_PTR pPrivateKeyTemplate, CK\_ULONG ulPrivateKeyAttributeCount, CK\_OBJECT\_HANDLE\_PTR phPublicKey, CK\_OBJECT\_HANDLE\_PTR phPrivateKey)  
*Generates a public-key/private-key pair using the specified mechanism.*
- **CK\_RV C\_WrapKey** (CK\_SESSION\_HANDLE hSession, CK\_MECHANISM\_PTR pMechanism, CK\_OBJECT\_HANDLE hWrappingKey, CK\_OBJECT\_HANDLE hKey, CK\_BYTE\_PTR pWrappedKey, CK\_ULONG\_PTR pulWrappedKeyLen)  
*Wraps (encrypts) the specified key using the specified wrapping key and mechanism.*
- **CK\_RV C\_UnwrapKey** (CK\_SESSION\_HANDLE hSession, CK\_MECHANISM\_PTR pMechanism, CK\_OBJECT\_HANDLE hUnwrappingKey, CK\_BYTE\_PTR pWrappedKey, CK\_ULONG ulWrappedKeyLen, CK\_ATTRIBUTE\_PTR pTemplate, CK\_ULONG ulCount, CK\_OBJECT\_HANDLE\_PTR phKey)  
*Unwraps (decrypts) the specified key using the specified unwrapping key.*
- **CK\_RV C\_DeriveKey** (CK\_SESSION\_HANDLE hSession, CK\_MECHANISM\_PTR pMechanism, CK\_OBJECT\_HANDLE hBaseKey, CK\_ATTRIBUTE\_PTR pTemplate, CK\_ULONG ulCount, CK\_OBJECT\_HANDLE\_PTR phKey)  
*Derive a key from the specified base key.*
- **CK\_RV C\_SeedRandom** (CK\_SESSION\_HANDLE hSession, CK\_BYTE\_PTR pSeed, CK\_ULONG ulSeedLen)  
*Mixes in additional seed material to the random number generator.*
- **CK\_RV C\_GenerateRandom** (CK\_SESSION\_HANDLE hSession, CK\_BYTE\_PTR pRandomData, CK\_ULONG ulRandomLen)  
*Generate the specified amount of random data.*
- **CK\_RV C\_GetFunctionStatus** (CK\_SESSION\_HANDLE hSession)  
*Legacy function - see PKCS#11 v2.40.*
- **CK\_RV C\_CancelFunction** (CK\_SESSION\_HANDLE hSession)  
*Legacy function.*
- **CK\_RV C\_WaitForSlotEvent** (CK\_FLAGS flags, CK\_SLOT\_ID\_PTR pSlot, CK\_VOID\_PTR pReserved)

*Wait for a slot event (token insertion, removal, etc) on the specified slot to occur.*

- `CK_RV pkcs11_mech_get_list (CK_SLOT_ID slotID, CK_MECHANISM_TYPE_PTR pMechanismList, CK_ULONG_PTR pulCount)`
- `CK_RV pkcs11_mech_get_info (CK_SLOT_ID slotID, CK_MECHANISM_TYPE type, CK_MECHANISM_INFO_PTR pInfo)`
- `CK_RV pkcs11_object_alloc (pkcs11_object_ptr *ppObject)`

**\*\***

- `CK_RV pkcs11_object_free (pkcs11_object_ptr pObject)`
- `CK_RV pkcs11_object_check (pkcs11_object_ptr *ppObject, CK_OBJECT_HANDLE hObject)`
- `CK_RV pkcs11_object_get_handle (pkcs11_object_ptr pObject, CK_OBJECT_HANDLE_PTR phObject)`
- `CK_RV pkcs11_object_get_name (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute)`
- `CK_RV pkcs11_object_get_class (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute)`
- `CK_RV pkcs11_object_get_type (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute)`
- `CK_RV pkcs11_object_get_destroyable (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute)`
- `CK_RV pkcs11_object_get_size (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject, CK_ULONG_PTR pulSize)`
- `CK_RV pkcs11_object_find (pkcs11_object_ptr *ppObject, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount)`
- `CK_RV pkcs11_object_create (CK_SESSION_HANDLE hSession, CK_ATTRIBUTE_PTR pTemplate, CK_ULONG ulCount, CK_OBJECT_HANDLE_PTR phObject)`

*Create a new object on the token in the specified session using the given attribute template.*

- `CK_RV pkcs11_object_destroy (CK_SESSION_HANDLE hSession, CK_OBJECT_HANDLE hObject)`

*Destroy the specified object.*

- `CK_RV pkcs11_object_deinit (pkcs11_lib_ctx_ptr pContext)`
- `CK_RV pkcs11_object_load_handle_info (pkcs11_lib_ctx_ptr pContext)`
- `CK_RV pkcs11_os_create_mutex (CK_VOID_PTR_PTR ppMutex)`

*Application callback for creating a mutex object.*

- `CK_RV pkcs11_os_destroy_mutex (CK_VOID_PTR pMutex)`
- `CK_RV pkcs11_os_lock_mutex (CK_VOID_PTR pMutex)`
- `CK_RV pkcs11_os_unlock_mutex (CK_VOID_PTR pMutex)`
- `pkcs11_session_ctx_ptr pkcs11_get_session_context (CK_SESSION_HANDLE hSession)`
- `CK_RV pkcs11_session_check (pkcs11_session_ctx_ptr *pSession, CK_SESSION_HANDLE hSession)`

*Check if the session is initialized properly.*

- `CK_RV pkcs11_session_open (CK_SLOT_ID slotID, CK_FLAGS flags, CK_VOID_PTR pApplication, CK_NOTIFY notify, CK_SESSION_HANDLE_PTR phSession)`
- `CK_RV pkcs11_session_close (CK_SESSION_HANDLE hSession)`
- `CK_RV pkcs11_session_closeall (CK_SLOT_ID slotID)`

*Close all sessions for a given slot - not actually all open sessions.*

- `CK_RV pkcs11_session_get_info (CK_SESSION_HANDLE hSession, CK_SESSION_INFO_PTR pInfo)`

*Obtains information about a particular session.*

- `CK_RV pkcs11_session_login (CK_SESSION_HANDLE hSession, CK_USER_TYPE userType, CK_UTF8CHAR_PTR pPin, CK_ULONG ulPinLen)`
- `CK_RV pkcs11_session_logout (CK_SESSION_HANDLE hSession)`
- `CK_RV pkcs11_signature_sign_init (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey)`

*Initialize a signing operation using the specified key and mechanism.*

- `CK_RV pkcs11_signature_sign (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen)`

*Sign the data in a single pass operation.*

- `CK_RV pkcs11_signature_sign_continue (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen)`

*Continues a multiple-part signature operation.*

- `CK_RV pkcs11_signature_sign_finish (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSignature, CK_ULONG_PTR pulSignatureLen)`



*Finishes a multiple-part signature operation.*

- `CK_RV pkcs11_signature_verify_init (CK_SESSION_HANDLE hSession, CK_MECHANISM_PTR pMechanism, CK_OBJECT_HANDLE hKey)`

*Initializes a verification operation using the specified key and mechanism.*

- `CK_RV pkcs11_signature_verify (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pData, CK_ULONG ulDataLen, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen)`

*Verifies a signature on single-part data.*

- `CK_RV pkcs11_signature_verify_continue (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pPart, CK_ULONG ulPartLen)`

*Continues a multiple-part verification operation.*

- `CK_RV pkcs11_signature_verify_finish (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pSignature, CK_ULONG ulSignatureLen)`

*Finishes a multiple-part verification operation.*

- `pkcs11_slot_ctx_ptr pkcs11_slot_get_context (pkcs11_lib_ctx_ptr lib_ctx, CK_SLOT_ID slotID)`

*Retrieve the current slot context.*

- `CK_VOID_PTR pkcs11_slot_initslots (CK_ULONG pulCount)`
- `CK_RV pkcs11_slot_config (CK_SLOT_ID slotID)`
- `CK_RV pkcs11_slot_init (CK_SLOT_ID slotID)`
- `CK_RV pkcs11_slot_get_list (CK_BBOOL tokenPresent, CK_SLOT_ID_PTR pSlotList, CK_ULONG_PTR pulCount)`
- `CK_RV pkcs11_slot_get_info (CK_SLOT_ID slotID, CK_SLOT_INFO_PTR pInfo)`

*Obtains information about a particular slot.*

- `CK_RV pkcs11_token_init (CK_SLOT_ID slotID, CK_UTF8CHAR_PTR pPin, CK_ULONG ulPinLen, CK_UTF8CHAR_PTR pLabel)`
- `CK_RV pkcs11_token_get_access_type (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute)`
- `CK_RV pkcs11_token_get_writable (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute)`
- `CK_RV pkcs11_token_get_storage (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute)`
- `CK_RV pkcs11_token_get_info (CK_SLOT_ID slotID, CK_TOKEN_INFO_PTR pInfo)`

*Obtains information about a particular token.*

- `CK_RV pkcs11_token_random (CK_SESSION_HANDLE hSession, CK_BYTE_PTR pRandomData, CK_ULONG ulRandomLen)`

*Generate the specified amount of random data.*

- `CK_RV pkcs11_token_convert_pin_to_key (const CK_UTF8CHAR_PTR pPin, const CK_ULONG ulPinLen, const CK_UTF8CHAR_PTR pSalt, const CK_ULONG ulSaltLen, CK_BYTE_PTR pKey, CK_ULONG ulKeyLen)`
- `CK_RV pkcs11_token_set_pin (CK_SESSION_HANDLE hSession, CK_UTF8CHAR_PTR pOldPin, CK_ULONG ulOldLen, CK_UTF8CHAR_PTR pNewPin, CK_ULONG ulNewLen)`
- `void pkcs11_util_escape_string (CK_UTF8CHAR_PTR buf, CK_ULONG buf_len)`
- `CK_RV pkcs11_util_convert_rv (ATCA_STATUS status)`
- `int pkcs11_util_memset (void *dest, size_t destsz, int ch, size_t count)`

## Variables

- `const pkcs11_attrb_model pkcs11_cert_x509public_attributes []`
- `const CK_ULONG pkcs11_cert_x509public_attributes_count = sizeof( pkcs11_cert_x509public_attributes ) / sizeof( pkcs11_cert_x509public_attributes [0])`
- `const pkcs11_attrb_model pkcs11_cert_wtlspublic_attributes []`
- `const CK_ULONG pkcs11_cert_wtlspublic_attributes_count = sizeof( pkcs11_cert_wtlspublic_attributes ) / sizeof( pkcs11_cert_wtlspublic_attributes [0])`
- `const pkcs11_attrb_model pkcs11_cert_x509_attributes []`
- `const CK_ULONG pkcs11_cert_x509_attributes_count = sizeof( pkcs11_cert_x509_attributes ) / sizeof( pkcs11_cert_x509_attributes [0])`
- `const char pkcs11_lib_manufacturer_id [] = "Microchip Technology Inc"`



- const char `pkcs11_lib_description` [] = "Cryptoauthlib PKCS11 Interface"
- const `pkcs11_attrib_model pkcs11_key_public_attributes` []
- const `CK_ULONG pkcs11_key_public_attributes_count` = sizeof( `pkcs11_key_public_attributes` ) / sizeof( `pkcs11_key_public_attributes` [0])
- const `pkcs11_attrib_model pkcs11_key_ec_public_attributes` []
- const `pkcs11_attrib_model pkcs11_key_private_attributes` []
- const `CK_ULONG pkcs11_key_private_attributes_count` = sizeof( `pkcs11_key_private_attributes` ) / sizeof( `pkcs11_key_private_attributes` [0])
- const `pkcs11_attrib_model pkcs11_key_rsa_private_attributes` []
- const `pkcs11_attrib_model pkcs11_key_ec_private_attributes` []
- const `pkcs11_attrib_model pkcs11_key_secret_attributes` []
- const `CK_ULONG pkcs11_key_secret_attributes_count` = sizeof( `pkcs11_key_secret_attributes` ) / sizeof( `pkcs11_key_secret_attributes` [0])
- `pkcs11_object_cache_t pkcs11_object_cache` [PKCS11\_MAX\_OBJECTS\_ALLOWED]
- const `pkcs11_attrib_model pkcs11_object_monotonic_attributes` []
- const `CK_ULONG pkcs11_object_monotonic_attributes_count` = sizeof( `pkcs11_object_monotonic_attributes` ) / sizeof( `pkcs11_object_monotonic_attributes` [0])

### 18.13.1 Detailed Description

### 18.13.2 Macro Definition Documentation

#### 18.13.2.1 PKCS11\_MECH\_ECC508\_EC\_CAPABILITY

```
#define PKCS11_MECH_ECC508_EC_CAPABILITY (CKF_EC_F_P | CKF_EC_NAMEDCURVE | CKF_EC_UNCOMPRESS)
```

#### 18.13.2.2 TABLE\_SIZE

```
#define TABLE_SIZE(  
    x ) sizeof(x) / sizeof(x[0])
```

### 18.13.3 Typedef Documentation

#### 18.13.3.1 pcks11\_mech\_table\_e

```
typedef struct _pcks11_mech_table_e pcks11_mech_table_e
```

### 18.13.3.2 pcks11\_mech\_table\_ptr

```
typedef struct _pcks11_mech_table_e * pcks11_mech_table_ptr
```

## 18.13.4 Function Documentation

### 18.13.4.1 C\_CancelFunction()

```
CK_RV C_CancelFunction (
    CK_SESSION_HANDLE hSession )
```

Legacy function.

### 18.13.4.2 C\_CloseAllSessions()

```
CK_RV C_CloseAllSessions (
    CK_SLOT_ID slotID )
```

Close all open sessions.

### 18.13.4.3 C\_CloseSession()

```
CK_RV C_CloseSession (
    CK_SESSION_HANDLE hSession )
```

Close the given session.

### 18.13.4.4 C\_CopyObject()

```
CK_RV C_CopyObject (
    CK_SESSION_HANDLE hSession,
    CK_OBJECT_HANDLE hObject,
    CK_ATTRIBUTE_PTR pTemplate,
    CK_ULONG ulCount,
    CK_OBJECT_HANDLE_PTR phNewObject )
```

Create a copy of the object with the specified handle.

#### 18.13.4.5 C\_CreateObject()

```
CK_RV C_CreateObject (
    CK_SESSION_HANDLE hSession,
    CK_ATTRIBUTE_PTR pTemplate,
    CK_ULONG ulCount,
    CK_OBJECT_HANDLE_PTR phObject )
```

Create a new object on the token in the specified session using the given attribute template.

#### 18.13.4.6 C\_Decrypt()

```
CK_RV C_Decrypt (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pEncryptedData,
    CK_ULONG ulEncryptedDataLen,
    CK_BYTE_PTR pData,
    CK_ULONG_PTR pulDataLen )
```

Perform a single operation decryption in the given session.

#### 18.13.4.7 C\_DecryptDigestUpdate()

```
CK_RV C_DecryptDigestUpdate (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pPart,
    CK_ULONG ulPartLen,
    CK_BYTE_PTR pDecryptedPart,
    CK_ULONG_PTR pulDecryptedPartLen )
```

Continues simultaneous multiple-part decryption and digesting operations.

#### 18.13.4.8 C\_DecryptFinal()

```
CK_RV C_DecryptFinal (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pData,
    CK_ULONG_PTR pDataLen )
```

Finishes a multiple-part decryption operation.

### 18.13.4.9 C\_DecryptInit()

```
CK_RV C_DecryptInit (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_OBJECT_HANDLE hObject )
```

Initialize decryption using the specified object.

### 18.13.4.10 C\_DecryptUpdate()

```
CK_RV C_DecryptUpdate (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pEncryptedData,
    CK_ULONG ulEncryptedDataLen,
    CK_BYTE_PTR pData,
    CK_ULONG_PTR pDataLen )
```

Continues a multiple-part decryption operation.

### 18.13.4.11 C\_DecryptVerifyUpdate()

```
CK_RV C_DecryptVerifyUpdate (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pEncryptedPart,
    CK_ULONG ulEncryptedPartLen,
    CK_BYTE_PTR pPart,
    CK_ULONG_PTR pulPartLen )
```

Continues simultaneous multiple-part decryption and verification operations.

### 18.13.4.12 C\_DeriveKey()

```
CK_RV C_DeriveKey (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_OBJECT_HANDLE hBaseKey,
    CK_ATTRIBUTE_PTR pTemplate,
    CK_ULONG ulCount,
    CK_OBJECT_HANDLE_PTR phKey )
```

Derive a key from the specified base key.

#### 18.13.4.13 C\_DestroyObject()

```
CK_RV C_DestroyObject (
    CK_SESSION_HANDLE hSession,
    CK_OBJECT_HANDLE hObject )
```

Destroy the specified object.

#### 18.13.4.14 C\_Digest()

```
CK_RV C_Digest (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pData,
    CK_ULONG ulDataLen,
    CK_BYTE_PTR pDigest,
    CK_ULONG_PTR pulDigestLen )
```

Digest the specified data in a one-pass operation and return the resulting digest.

#### 18.13.4.15 C\_DigestEncryptUpdate()

```
CK_RV C_DigestEncryptUpdate (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pPart,
    CK_ULONG ulPartLen,
    CK_BYTE_PTR pEncryptedPart,
    CK_ULONG_PTR pulEncryptedPartLen )
```

Continues simultaneous multiple-part digesting and encryption operations.

#### 18.13.4.16 C\_DigestFinal()

```
CK_RV C_DigestFinal (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pDigest,
    CK_ULONG_PTR pulDigestLen )
```

Finishes a multiple-part digesting operation.

### 18.13.4.17 C\_DigestInit()

```
CK_RV C_DigestInit (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism )
```

Initializes a message-digesting operation using the specified mechanism in the specified session.

### 18.13.4.18 C\_DigestKey()

```
CK_RV C_DigestKey (
    CK_SESSION_HANDLE hSession,
    CK_OBJECT_HANDLE hObject )
```

Update a running digest operation by digesting a secret key with the specified handle.

### 18.13.4.19 C\_DigestUpdate()

```
CK_RV C_DigestUpdate (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pPart,
    CK_ULONG ulPartLen )
```

Continues a multiple-part digesting operation.

### 18.13.4.20 C\_Encrypt()

```
CK_RV C_Encrypt (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pData,
    CK_ULONG ulDataLen,
    CK_BYTE_PTR pEncryptedData,
    CK_ULONG_PTR pulEncryptedDataLen )
```

Perform a single operation encryption operation in the specified session.

### 18.13.4.21 C\_EncryptFinal()

```
CK_RV C_EncryptFinal (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pEncryptedData,
    CK_ULONG_PTR pulEncryptedDataLen )
```

Finishes a multiple-part encryption operation.

#### 18.13.4.22 C\_EncryptInit()

```
CK_RV C_EncryptInit (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_OBJECT_HANDLE hObject )
```

Initializes an encryption operation using the specified mechanism and session.

#### 18.13.4.23 C\_EncryptUpdate()

```
CK_RV C_EncryptUpdate (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pData,
    CK_ULONG ulDataLen,
    CK_BYTE_PTR pEncryptedData,
    CK_ULONG_PTR pulEncryptedDataLen )
```

Continues a multiple-part encryption operation.

#### 18.13.4.24 C\_Finalize()

```
CK_RV C_Finalize (
    CK_VOID_PTR pReserved )
```

Clean up miscellaneous Cryptoki-associated resources.

#### 18.13.4.25 C\_FindObjects()

```
CK_RV C_FindObjects (
    CK_SESSION_HANDLE hSession,
    CK_OBJECT_HANDLE_PTR phObject,
    CK_ULONG ulMaxObjectCount,
    CK_ULONG_PTR pulObjectCount )
```

Continue the search for objects in the specified session.

#### 18.13.4.26 C\_FindObjectsFinal()

```
CK_RV C_FindObjectsFinal (
    CK_SESSION_HANDLE hSession )
```

Finishes an object search operation (and cleans up)

### 18.13.4.27 C\_FindObjectsInit()

```
CK_RV C_FindObjectsInit (
    CK_SESSION_HANDLE hSession,
    CK_ATTRIBUTE_PTR pTemplate,
    CK_ULONG ulCount )
```

Initializes an object search in the specified session using the specified attribute template as search parameters.

### 18.13.4.28 C\_GenerateKey()

```
CK_RV C_GenerateKey (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_ATTRIBUTE_PTR pTemplate,
    CK_ULONG ulCount,
    CK_OBJECT_HANDLE_PTR phKey )
```

Generates a secret key using the specified mechanism.

### 18.13.4.29 C\_GenerateKeyPair()

```
CK_RV C_GenerateKeyPair (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_ATTRIBUTE_PTR pPublicKeyTemplate,
    CK_ULONG ulPublicKeyAttributeCount,
    CK_ATTRIBUTE_PTR pPrivateKeyTemplate,
    CK_ULONG ulPrivateKeyAttributeCount,
    CK_OBJECT_HANDLE_PTR phPublicKey,
    CK_OBJECT_HANDLE_PTR phPrivateKey )
```

Generates a public-key/private-key pair using the specified mechanism.

### 18.13.4.30 C\_GenerateRandom()

```
CK_RV C_GenerateRandom (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pRandomData,
    CK_ULONG ulRandomLen )
```

Generate the specified amount of random data.



**18.13.4.31 C\_GetAttributeValue()**

```
CK_RV C_GetAttributeValue (
    CK_SESSION_HANDLE hSession,
    CK_OBJECT_HANDLE hObject,
    CK_ATTRIBUTE_PTR pTemplate,
    CK_ULONG ulCount )
```

Obtains an attribute value of an object.

**18.13.4.32 C\_GetFunctionList()**

```
CK_RV C_GetFunctionList (
    CK_FUNCTION_LIST_PTR_PTR ppFunctionList )
```

Obtains entry points of Cryptoki library functions.

**18.13.4.33 C\_GetFunctionStatus()**

```
CK_RV C_GetFunctionStatus (
    CK_SESSION_HANDLE hSession )
```

Legacy function - see PKCS#11 v2.40.

**18.13.4.34 C\_GetInfo()**

```
CK_RV C_GetInfo (
    CK_INFO_PTR pInfo )
```

Obtains general information about Cryptoki.

**18.13.4.35 C\_GetMechanismInfo()**

```
CK_RV C_GetMechanismInfo (
    CK_SLOT_ID slotID,
    CK_MECHANISM_TYPE type,
    CK_MECHANISM_INFO_PTR pInfo )
```

Obtains information about a particular mechanism of a token (in a slot)

### 18.13.4.36 C\_GetMechanismList()

```
CK_RV C_GetMechanismList (
    CK_SLOT_ID slotID,
    CK_MECHANISM_TYPE_PTR pMechanismList,
    CK_ULONG_PTR pulCount )
```

Obtains a list of mechanisms supported by a token (in a slot)

### 18.13.4.37 C\_GetObjectSize()

```
CK_RV C_GetObjectSize (
    CK_SESSION_HANDLE hSession,
    CK_OBJECT_HANDLE hObject,
    CK_ULONG_PTR pulSize )
```

Obtains the size of an object in bytes.

### 18.13.4.38 C\_GetOperationState()

```
CK_RV C_GetOperationState (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pOperationState,
    CK_ULONG_PTR pulOperationStateLen )
```

Obtains the cryptographic operations state of a session.

### 18.13.4.39 C\_GetSessionInfo()

```
CK_RV C_GetSessionInfo (
    CK_SESSION_HANDLE hSession,
    CK_SESSION_INFO_PTR pInfo )
```

Retrieve information about the specified session.

### 18.13.4.40 C\_GetSlotInfo()

```
CK_RV C_GetSlotInfo (
    CK_SLOT_ID slotID,
    CK_SLOT_INFO_PTR pInfo )
```

Obtains information about a particular slot.

#### 18.13.4.41 C\_GetSlotList()

```
CK_RV C_GetSlotList (
    CK_BBOOL tokenPresent,
    CK_SLOT_ID_PTR pSlotList,
    CK_ULONG_PTR pulCount )
```

Obtains a list of slots in the system.

#### 18.13.4.42 C\_GetTokenInfo()

```
CK_RV C_GetTokenInfo (
    CK_SLOT_ID slotID,
    CK_TOKEN_INFO_PTR pInfo )
```

Obtains information about a particular token.

#### 18.13.4.43 C\_Initialize()

```
CK_RV C_Initialize (
    CK_VOID_PTR pInitArgs )
```

Initializes Cryptoki library NOTES: If pInitArgs is a non-NULL\_PTR is must dereference to a [CK\\_C\\_INITIALIZE\\_ARGS](#) structure.

#### 18.13.4.44 C\_InitPIN()

```
CK_RV C_InitPIN (
    CK_SESSION_HANDLE hSession,
    CK_UTF8CHAR_PTR pPin,
    CK_ULONG ulPinLen )
```

Initializes the normal user's PIN.

#### 18.13.4.45 C\_InitToken()

```
CK_RV C_InitToken (
    CK_SLOT_ID slotID,
    CK_UTF8CHAR_PTR pPin,
    CK_ULONG ulPinLen,
    CK_UTF8CHAR_PTR pLabel )
```

Initializes a token (in a slot)

### 18.13.4.46 C\_Login()

```
CK_RV C_Login (
    CK_SESSION_HANDLE hSession,
    CK_USER_TYPE userType,
    CK_UTF8CHAR_PTR pPin,
    CK_ULONG ulPinLen )
```

Login on the token in the specified session.

### 18.13.4.47 C\_Logout()

```
CK_RV C_Logout (
    CK_SESSION_HANDLE hSession )
```

Log out of the token in the specified session.

### 18.13.4.48 C\_OpenSession()

```
CK_RV C_OpenSession (
    CK_SLOT_ID slotID,
    CK_FLAGS flags,
    CK_VOID_PTR pApplication,
    CK_NOTIFY notify,
    CK_SESSION_HANDLE_PTR phSession )
```

Opens a connection between an application and a particular token or sets up an application callback for token insertion.

### 18.13.4.49 C\_SeedRandom()

```
CK_RV C_SeedRandom (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pSeed,
    CK_ULONG ulSeedLen )
```

Mixes in additional seed material to the random number generator.

#### 18.13.4.50 C\_SetAttributeValue()

```
CK_RV C_SetAttributeValue (
    CK_SESSION_HANDLE hSession,
    CK_OBJECT_HANDLE hObject,
    CK_ATTRIBUTE_PTR pTemplate,
    CK_ULONG ulCount )
```

Change or set the value of the specified attributes on the specified object.

#### 18.13.4.51 C\_SetOperationState()

```
CK_RV C_SetOperationState (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pOperationState,
    CK_ULONG ulOperationStateLen,
    CK_OBJECT_HANDLE hEncryptionKey,
    CK_OBJECT_HANDLE hAuthenticationKey )
```

Sets the cryptographic operations state of a session.

#### 18.13.4.52 C\_SetPIN()

```
CK_RV C_SetPIN (
    CK_SESSION_HANDLE hSession,
    CK_UTF8CHAR_PTR pOldPin,
    CK_ULONG ulOldLen,
    CK_UTF8CHAR_PTR pNewPin,
    CK_ULONG ulNewLen )
```

Modifies the PIN of the current user.

#### 18.13.4.53 C\_Sign()

```
CK_RV C_Sign (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pData,
    CK_ULONG ulDataLen,
    CK_BYTE_PTR pSignature,
    CK_ULONG_PTR pulSignatureLen )
```

Sign the data in a single pass operation.

### 18.13.4.54 C\_SignEncryptUpdate()

```
CK_RV C_SignEncryptUpdate (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pPart,
    CK_ULONG ulPartLen,
    CK_BYTE_PTR pEncryptedPart,
    CK_ULONG_PTR pulEncryptedPartLen )
```

Continues simultaneous multiple-part signature and encryption operations.

### 18.13.4.55 C\_SignFinal()

```
CK_RV C_SignFinal (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pSignature,
    CK_ULONG_PTR pulSignatureLen )
```

Finishes a multiple-part signature operation.

### 18.13.4.56 C\_SignInit()

```
CK_RV C_SignInit (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_OBJECT_HANDLE hKey )
```

Initialize a signing operation using the specified key and mechanism.

### 18.13.4.57 C\_SignRecover()

```
CK_RV C_SignRecover (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pData,
    CK_ULONG ulDataLen,
    CK_BYTE_PTR pSignature,
    CK_ULONG_PTR pulSignatureLen )
```

Signs single-part data, where the data can be recovered from the signature.

#### 18.13.4.58 C\_SignRecoverInit()

```
CK_RV C_SignRecoverInit (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_OBJECT_HANDLE hKey )
```

Initializes a signature operation, where the data can be recovered from the signature.

#### 18.13.4.59 C\_SignUpdate()

```
CK_RV C_SignUpdate (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pPart,
    CK_ULONG ulPartLen )
```

Continues a multiple-part signature operation.

#### 18.13.4.60 C\_UnwrapKey()

```
CK_RV C_UnwrapKey (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_OBJECT_HANDLE hUnwrappingKey,
    CK_BYTE_PTR pWrappedKey,
    CK_ULONG ulWrappedKeyLen,
    CK_ATTRIBUTE_PTR pTemplate,
    CK_ULONG ulCount,
    CK_OBJECT_HANDLE_PTR phKey )
```

Unwraps (decrypts) the specified key using the specified unwrapping key.

#### 18.13.4.61 C\_Verify()

```
CK_RV C_Verify (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pData,
    CK_ULONG ulDataLen,
    CK_BYTE_PTR pSignature,
    CK_ULONG ulSignatureLen )
```

Verifies a signature on single-part data.

### 18.13.4.62 C\_VerifyFinal()

```
CK_RV C_VerifyFinal (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pSignature,
    CK_ULONG ulSignatureLen )
```

Finishes a multiple-part verification operation.

### 18.13.4.63 C\_VerifyInit()

```
CK_RV C_VerifyInit (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_OBJECT_HANDLE hKey )
```

Initializes a verification operation using the specified key and mechanism.

### 18.13.4.64 C\_VerifyRecover()

```
CK_RV C_VerifyRecover (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pSignature,
    CK_ULONG ulSignatureLen,
    CK_BYTE_PTR pData,
    CK_ULONG_PTR pulDataLen )
```

Verifies a signature on single-part data, where the data is recovered from the signature.

### 18.13.4.65 C\_VerifyRecoverInit()

```
CK_RV C_VerifyRecoverInit (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_OBJECT_HANDLE hKey )
```

Initializes a verification operation where the data is recovered from the signature.

### 18.13.4.66 C\_VerifyUpdate()

```
CK_RV C_VerifyUpdate (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pPart,
    CK_ULONG ulPartLen )
```

Continues a multiple-part verification operation.



**18.13.4.67 C\_WaitForSlotEvent()**

```
CK_RV C_WaitForSlotEvent (
    CK_FLAGS flags,
    CK_SLOT_ID_PTR pSlot,
    CK_VOID_PTR pReserved )
```

Wait for a slot event (token insertion, removal, etc) on the specified slot to occur.

**18.13.4.68 C\_WrapKey()**

```
CK_RV C_WrapKey (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_OBJECT_HANDLE hWrappingKey,
    CK_OBJECT_HANDLE hKey,
    CK_BYTE_PTR pWrappedKey,
    CK_ULONG_PTR pulWrappedKeyLen )
```

Wraps (encrypts) the specified key using the specified wrapping key and mechanism.

**18.13.4.69 pkcs11\_attrib\_empty()**

```
CK_RV pkcs11_attrib_empty (
    const CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

**18.13.4.70 pkcs11\_attrib\_false()**

```
CK_RV pkcs11_attrib_false (
    const CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

**18.13.4.71 pkcs11\_attrib\_fill()**

```
CK_RV pkcs11_attrib_fill (
    CK_ATTRIBUTE_PTR pAttribute,
    const CK_VOID_PTR pData,
    const CK_ULONG ulSize )
```

Perform the nessasary checks and copy data into an attribute structure.

The ulValueLen field is modified to hold the exact length of the specified attribute for the object. In the special case of an attribute whose value is an array of attributes, for example CKA\_WRAP\_TEMPLATE, where it is passed in with pValue not NULL, then if the pValue of elements within the array is NULL\_PTR then the ulValueLen of elements within the array will be set to the required length. If the pValue of elements within the array is not NULL\_PTR, then the ulValueLen element of attributes within the array MUST reflect the space that the corresponding pValue points to, and pValue is filled in if there is sufficient room. Therefore it is important to initialize the contents of a buffer before calling C\_GetAttributeValue to get such an array value. If any ulValueLen within the array isn't large enough, it will be set to CK\_UNAVAILABLE\_INFORMATION and the function will return CKR\_BUFFER\_TOO\_SMALL, as it does if an attribute in the pTemplate argument has ulValueLen too small. Note that any attribute whose value is an array of attributes is identifiable by virtue of the attribute type having the CKF\_ARRAY\_ATTRIBUTE bit set.

### 18.13.4.72 pkcs11\_attrib\_true()

```
CK_RV pkcs11_attrib_true (
    const CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

### 18.13.4.73 pkcs11\_attrib\_value()

```
CK_RV pkcs11_attrib_value (
    CK_ATTRIBUTE_PTR pAttribute,
    const CK_ULONG ulValue,
    const CK_ULONG ulSize )
```

Helper function to write a numerical value to an attribute buffer.

### 18.13.4.74 pkcs11\_cert\_get\_authority\_key\_id()

```
CK_RV pkcs11_cert_get_authority_key_id (
    CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

### 18.13.4.75 pkcs11\_cert\_get\_encoded()

```
CK_RV pkcs11_cert_get_encoded (
    CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

### 18.13.4.76 pkcs11\_cert\_get\_subject()

```
CK_RV pkcs11_cert_get_subject (
    CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

### 18.13.4.77 pkcs11\_cert\_get\_subject\_key\_id()

```
CK_RV pkcs11_cert_get_subject_key_id (
    CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

**18.13.4.78 pkcs11\_cert\_get\_trusted\_flag()**

```
CK_RV pkcs11_cert_get_trusted_flag (
    CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

**18.13.4.79 pkcs11\_cert\_get\_type()**

```
CK_RV pkcs11_cert_get_type (
    CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

**18.13.4.80 pkcs11\_cert\_x509\_write()**

```
CK_RV pkcs11_cert_x509_write (
    CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

**18.13.4.81 pkcs11\_config\_cert()**

```
CK_RV pkcs11_config_cert (
    pkcs11_lib_ctx_ptr pLibCtx,
    pkcs11_slot_ctx_ptr pSlot,
    pkcs11_object_ptr pObject,
    CK_ATTRIBUTE_PTR pLabel )
```

**18.13.4.82 pkcs11\_config\_init\_cert()**

```
void pkcs11_config_init_cert (
    pkcs11_object_ptr pObject,
    char * label,
    size_t len )
```

**18.13.4.83 pkcs11\_config\_init\_private()**

```
void pkcs11_config_init_private (
    pkcs11_object_ptr pObject,
    char * label,
    size_t len )
```

### 18.13.4.84 pkcs11\_config\_init\_public()

```
void pkcs11_config_init_public (
    pkcs11_object_ptr pObject,
    char * label,
    size_t len )
```

### 18.13.4.85 pkcs11\_config\_key()

```
CK_RV pkcs11_config_key (
    pkcs11_lib_ctx_ptr pLibCtx,
    pkcs11_slot_ctx_ptr pSlot,
    pkcs11_object_ptr pObject,
    CK_ATTRIBUTE_PTR pLabel )
```

### 18.13.4.86 pkcs11\_config\_load()

```
CK_RV pkcs11_config_load (
    pkcs11_slot_ctx_ptr slot_ctx )
```

### 18.13.4.87 pkcs11\_config\_load\_objects()

```
CK_RV pkcs11_config_load_objects (
    pkcs11_slot_ctx_ptr slot_ctx )
```

### 18.13.4.88 pkcs11\_config\_remove\_object()

```
CK_RV pkcs11_config_remove_object (
    pkcs11_lib_ctx_ptr pLibCtx,
    pkcs11_slot_ctx_ptr pSlot,
    pkcs11_object_ptr pObject )
```

### 18.13.4.89 pkcs11\_deinit()

```
CK_RV pkcs11_deinit (
    CK_VOID_PTR pReserved )
```

**Todo** If other threads are waiting for something to happen this call should cause those calls to unblock and return CKR\_CRYPTOKI\_NOT\_INITIALIZED - How that is done by this simplified mutex API is yet to be determined

**18.13.4.90 pkcs11\_find\_continue()**

```
CK_RV pkcs11_find_continue (
    CK_SESSION_HANDLE hSession,
    CK_OBJECT_HANDLE_PTR phObject,
    CK_ULONG ulMaxObjectCount,
    CK_ULONG_PTR pulObjectCount )
```

**18.13.4.91 pkcs11\_find\_finish()**

```
CK_RV pkcs11_find_finish (
    CK_SESSION_HANDLE hSession )
```

**18.13.4.92 pkcs11\_find\_get\_attribute()**

```
CK_RV pkcs11_find_get_attribute (
    CK_SESSION_HANDLE hSession,
    CK_OBJECT_HANDLE hObject,
    CK_ATTRIBUTE_PTR pTemplate,
    CK_ULONG ulCount )
```

**18.13.4.93 pkcs11\_find\_init()**

```
CK_RV pkcs11_find_init (
    CK_SESSION_HANDLE hSession,
    CK_ATTRIBUTE_PTR pTemplate,
    CK_ULONG ulCount )
```

**18.13.4.94 pkcs11\_get\_context()**

```
pkcs11_lib_ctx_ptr pkcs11_get_context (
    void )
```

Retrieve the current library context.

**18.13.4.95 pkcs11\_get\_lib\_info()**

```
CK_RV pkcs11_get_lib_info (
    CK_INFO_PTR pInfo )
```

Obtains general information about Cryptoki.

### 18.13.4.96 pkcs11\_get\_session\_context()

```
pkcs11_session_ctx_ptr pkcs11_get_session_context (
    CK_SESSION_HANDLE hSession )
```

### 18.13.4.97 pkcs11\_init()

```
CK_RV pkcs11_init (
    CK_C_INITIALIZE_ARGS_PTR pInitArgs )
```

Initializes the PKCS11 API Library for Cryptoauthlib.

**Todo** This is where we should allocate a new context if we're using dynamic memory

**Todo** If we're using dynamic memory we need to make sure to deallocate it if any of the errors after the allocations are encountered

### 18.13.4.98 pkcs11\_init\_check()

```
CK_RV pkcs11_init_check (
    pkcs11_lib_ctx_ptr * ppContext,
    CK_BBOOL lock )
```

Check if the library is initialized properly.

### 18.13.4.99 pkcs11\_key\_derive()

```
CK_RV pkcs11_key_derive (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_OBJECT_HANDLE hBaseKey,
    CK_ATTRIBUTE_PTR pTemplate,
    CK_ULONG ulCount,
    CK_OBJECT_HANDLE_PTR phKey )
```

**18.13.4.100 pkcs11\_key\_generate\_pair()**

```
CK_RV pkcs11_key_generate_pair (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_ATTRIBUTE_PTR pPublicKeyTemplate,
    CK_ULONG ulPublicKeyAttributeCount,
    CK_ATTRIBUTE_PTR pPrivateKeyTemplate,
    CK_ULONG ulPrivateKeyAttributeCount,
    CK_OBJECT_HANDLE_PTR phPublicKey,
    CK_OBJECT_HANDLE_PTR phPrivateKey )
```

**18.13.4.101 pkcs11\_key\_write()**

```
CK_RV pkcs11_key_write (
    CK_VOID_PTR pSession,
    CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

**18.13.4.102 pkcs11\_lock\_context()**

```
CK_RV pkcs11_lock_context (
    pkcs11_lib_ctx_ptr pContext )
```

**18.13.4.103 pkcs11\_mech\_get\_list()**

```
CK_RV pkcs11_mech_get_list (
    CK_SLOT_ID slotID,
    CK_MECHANISM_TYPE_PTR pMechanismList,
    CK_ULONG_PTR pulCount )
```

**18.13.4.104 pkcs11\_object\_alloc()**

```
CK_RV pkcs11_object_alloc (
    pkcs11_object_ptr * ppObject )
```

\*\*

\*\*

### 18.13.4.105 pkcs11\_object\_check()

```
CK_RV pkcs11_object_check (
    pkcs11_object_ptr * ppObject,
    CK_OBJECT_HANDLE hObject )
```

### 18.13.4.106 pkcs11\_object\_create()

```
CK_RV pkcs11_object_create (
    CK_SESSION_HANDLE hSession,
    CK_ATTRIBUTE_PTR pTemplate,
    CK_ULONG ulCount,
    CK_OBJECT_HANDLE_PTR phObject )
```

Create a new object on the token in the specified session using the given attribute template.

### 18.13.4.107 pkcs11\_object\_deinit()

```
CK_RV pkcs11_object_deinit (
    pkcs11_lib_ctx_ptr pContext )
```

### 18.13.4.108 pkcs11\_object\_destroy()

```
CK_RV pkcs11_object_destroy (
    CK_SESSION_HANDLE hSession,
    CK_OBJECT_HANDLE hObject )
```

Destroy the specified object.

### 18.13.4.109 pkcs11\_object\_find()

```
CK_RV pkcs11_object_find (
    pkcs11_object_ptr * ppObject,
    CK_ATTRIBUTE_PTR pTemplate,
    CK_ULONG ulCount )
```

### 18.13.4.110 pkcs11\_object\_free()

```
CK_RV pkcs11_object_free (
    pkcs11_object_ptr pObject )
```



**18.13.4.111 pkcs11\_object\_get\_class()**

```
CK_RV pkcs11_object_get_class (
    CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

**18.13.4.112 pkcs11\_object\_get\_destroyable()**

```
CK_RV pkcs11_object_get_destroyable (
    CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

**18.13.4.113 pkcs11\_object\_get\_handle()**

```
CK_RV pkcs11_object_get_handle (
    pkcs11_object_ptr pObject,
    CK_OBJECT_HANDLE_PTR phObject )
```

**18.13.4.114 pkcs11\_object\_get\_name()**

```
CK_RV pkcs11_object_get_name (
    CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

**18.13.4.115 pkcs11\_object\_get\_size()**

```
CK_RV pkcs11_object_get_size (
    CK_SESSION_HANDLE hSession,
    CK_OBJECT_HANDLE hObject,
    CK_ULONG_PTR pulSize )
```

**18.13.4.116 pkcs11\_object\_get\_type()**

```
CK_RV pkcs11_object_get_type (
    CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

### 18.13.4.117 pkcs11\_object\_load\_handle\_info()

```
CK_RV pkcs11_object_load_handle_info (
    pkcs11_lib_ctx_ptr pContext )
```

### 18.13.4.118 pkcs11\_os\_create\_mutex()

```
CK_RV pkcs11_os_create_mutex (
    CK_VOID_PTR_PTR ppMutex )
```

Application callback for creating a mutex object.

#### Parameters

in, out	<i>ppMutex</i>	location to receive ptr to mutex
---------	----------------	----------------------------------

### 18.13.4.119 pkcs11\_os\_destroy\_mutex()

```
CK_RV pkcs11_os_destroy_mutex (
    CK_VOID_PTR pMutex )
```

### 18.13.4.120 pkcs11\_os\_lock\_mutex()

```
CK_RV pkcs11_os_lock_mutex (
    CK_VOID_PTR pMutex )
```

### 18.13.4.121 pkcs11\_os\_unlock\_mutex()

```
CK_RV pkcs11_os_unlock_mutex (
    CK_VOID_PTR pMutex )
```

### 18.13.4.122 pkcs11\_session\_check()

```
CK_RV pkcs11_session_check (
    pkcs11_session_ctx_ptr * pSession,
    CK_SESSION_HANDLE hSession )
```

Check if the session is initialized properly.

**18.13.4.123 pkcs11\_session\_close()**

```
CK_RV pkcs11_session_close (
    CK_SESSION_HANDLE hSession )
```

**18.13.4.124 pkcs11\_session\_closeall()**

```
CK_RV pkcs11_session_closeall (
    CK_SLOT_ID slotID )
```

Close all sessions for a given slot - not actually all open sessions.

**18.13.4.125 pkcs11\_session\_get\_info()**

```
CK_RV pkcs11_session_get_info (
    CK_SESSION_HANDLE hSession,
    CK_SESSION_INFO_PTR pInfo )
```

Obtains information about a particular session.

**18.13.4.126 pkcs11\_session\_login()**

```
CK_RV pkcs11_session_login (
    CK_SESSION_HANDLE hSession,
    CK_USER_TYPE userType,
    CK_UTF8CHAR_PTR pPin,
    CK_ULONG ulPinLen )
```

**18.13.4.127 pkcs11\_session\_logout()**

```
CK_RV pkcs11_session_logout (
    CK_SESSION_HANDLE hSession )
```

**18.13.4.128 pkcs11\_session\_open()**

```
CK_RV pkcs11_session_open (
    CK_SLOT_ID slotID,
    CK_FLAGS flags,
    CK_VOID_PTR pApplication,
    CK_NOTIFY notify,
    CK_SESSION_HANDLE_PTR phSession )
```

### 18.13.4.129 pkcs11\_signature\_sign()

```
CK_RV pkcs11_signature_sign (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pData,
    CK_ULONG ulDataLen,
    CK_BYTE_PTR pSignature,
    CK_ULONG_PTR pulSignatureLen )
```

Sign the data in a single pass operation.

### 18.13.4.130 pkcs11\_signature\_sign\_continue()

```
CK_RV pkcs11_signature_sign_continue (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pPart,
    CK_ULONG ulPartLen )
```

Continues a multiple-part signature operation.

### 18.13.4.131 pkcs11\_signature\_sign\_finish()

```
CK_RV pkcs11_signature_sign_finish (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pSignature,
    CK_ULONG_PTR pulSignatureLen )
```

Finishes a multiple-part signature operation.

### 18.13.4.132 pkcs11\_signature\_sign\_init()

```
CK_RV pkcs11_signature_sign_init (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_OBJECT_HANDLE hKey )
```

Initialize a signing operation using the specified key and mechanism.

**18.13.4.133 pkcs11\_signature\_verify()**

```
CK_RV pkcs11_signature_verify (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pData,
    CK_ULONG ulDataLen,
    CK_BYTE_PTR pSignature,
    CK_ULONG ulSignatureLen )
```

Verifies a signature on single-part data.

**18.13.4.134 pkcs11\_signature\_verify\_continue()**

```
CK_RV pkcs11_signature_verify_continue (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pPart,
    CK_ULONG ulPartLen )
```

Continues a multiple-part verification operation.

**18.13.4.135 pkcs11\_signature\_verify\_finish()**

```
CK_RV pkcs11_signature_verify_finish (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pSignature,
    CK_ULONG ulSignatureLen )
```

Finishes a multiple-part verification operation.

**18.13.4.136 pkcs11\_signature\_verify\_init()**

```
CK_RV pkcs11_signature_verify_init (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism,
    CK_OBJECT_HANDLE hKey )
```

Initializes a verification operation using the specified key and mechanism.

**18.13.4.137 pkcs11\_slot\_config()**

```
CK_RV pkcs11_slot_config (
    CK_SLOT_ID slotID )
```

### 18.13.4.138 pkcs11\_slot\_get\_context()

```
pkcs11_slot_ctx_ptr pkcs11_slot_get_context (
    pkcs11_lib_ctx_ptr lib_ctx,
    CK_SLOT_ID slotID )
```

Retrieve the current slot context.

### 18.13.4.139 pkcs11\_slot\_get\_info()

```
CK_RV pkcs11_slot_get_info (
    CK_SLOT_ID slotID,
    CK_SLOT_INFO_PTR pInfo )
```

Obtains information about a particular slot.

### 18.13.4.140 pkcs11\_slot\_get\_list()

```
CK_RV pkcs11_slot_get_list (
    CK_BBOOL tokenPresent,
    CK_SLOT_ID_PTR pSlotList,
    CK_ULONG_PTR pulCount )
```

### 18.13.4.141 pkcs11\_slot\_init()

```
CK_RV pkcs11_slot_init (
    CK_SLOT_ID slotID )
```

### 18.13.4.142 pkcs11\_slot\_initslots()

```
CK_VOID_PTR pkcs11_slot_initslots (
    CK_ULONG pulCount )
```

### 18.13.4.143 pkcs11\_token\_convert\_pin\_to\_key()

```
CK_RV pkcs11_token_convert_pin_to_key (
    const CK_UTF8CHAR_PTR pPin,
    const CK_ULONG ulPinLen,
    const CK_UTF8CHAR_PTR pSalt,
    const CK_ULONG ulSaltLen,
    CK_BYTE_PTR pKey,
    CK_ULONG ulKeyLen )
```

**18.13.4.144 pkcs11\_token\_get\_access\_type()**

```
CK_RV pkcs11_token_get_access_type (
    CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

**18.13.4.145 pkcs11\_token\_get\_info()**

```
CK_RV pkcs11_token_get_info (
    CK_SLOT_ID slotID,
    CK_TOKEN_INFO_PTR pInfo )
```

Obtains information about a particular token.

**18.13.4.146 pkcs11\_token\_get\_storage()**

```
CK_RV pkcs11_token_get_storage (
    CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

**18.13.4.147 pkcs11\_token\_get\_writable()**

```
CK_RV pkcs11_token_get_writable (
    CK_VOID_PTR pObject,
    CK_ATTRIBUTE_PTR pAttribute )
```

**18.13.4.148 pkcs11\_token\_init()**

```
CK_RV pkcs11_token_init (
    CK_SLOT_ID slotID,
    CK_UTF8CHAR_PTR pPin,
    CK_ULONG ulPinLen,
    CK_UTF8CHAR_PTR pLabel )
```

Write the configuration into the device and generate new keys

**18.13.4.149 pkcs11\_token\_random()**

```
CK_RV pkcs11_token_random (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pRandomData,
    CK_ULONG ulRandomLen )
```

Generate the specified amount of random data.

### 18.13.4.150 pkcs11\_token\_set\_pin()

```
CK_RV pkcs11_token_set_pin (
    CK_SESSION_HANDLE hSession,
    CK_UTF8CHAR_PTR pOldPin,
    CK_ULONG ulOldLen,
    CK_UTF8CHAR_PTR pNewPin,
    CK_ULONG ulNewLen )
```

### 18.13.4.151 pkcs11\_unlock\_context()

```
CK_RV pkcs11_unlock_context (
    pkcs11_lib_ctx_ptr pContext )
```

### 18.13.4.152 pkcs11\_util\_convert\_rv()

```
CK_RV pkcs11_util_convert_rv (
    ATCA_STATUS status )
```

### 18.13.4.153 pkcs11\_util\_escape\_string()

```
void pkcs11_util_escape_string (
    CK_UTF8CHAR_PTR buf,
    CK_ULONG buf_len )
```

### 18.13.4.154 pkcs11\_util\_memset()

```
int pkcs11_util_memset (
    void * dest,
    size_t destsz,
    int ch,
    size_t count )
```

### 18.13.4.155 pkcs\_mech\_get\_info()

```
CK_RV pkcs_mech_get_info (
    CK_SLOT_ID slotID,
    CK_MECHANISM_TYPE type,
    CK_MECHANISM_INFO_PTR pInfo )
```



## 18.13.5 Variable Documentation

### 18.13.5.1 pkcs11\_cert\_wtlspublic\_attributes

```
const pkcs11_attr_model pkcs11_cert_wtlspublic_attributes[]
```

CKO\_CERTIFICATE (Type: CKC\_WTLS) - TLS Public Key Certificate Model

### 18.13.5.2 pkcs11\_cert\_wtlspublic\_attributes\_count

```
const CK_ULONG pkcs11_cert_wtlspublic_attributes_count = sizeof( pkcs11_cert_wtlspublic_attributes ) / sizeof( pkcs11_cert_wtlspublic_attributes [0])
```

### 18.13.5.3 pkcs11\_cert\_x509\_attributes

```
const pkcs11_attr_model pkcs11_cert_x509_attributes[]
```

CKO\_CERTIFICATE (Type: CKC\_X\_509\_ATTR\_CERT) - X509 Attribute Certificate Model

### 18.13.5.4 pkcs11\_cert\_x509\_attributes\_count

```
const CK_ULONG pkcs11_cert_x509_attributes_count = sizeof( pkcs11_cert_x509_attributes ) / sizeof( pkcs11_cert_x509_attributes [0])
```

### 18.13.5.5 pkcs11\_cert\_x509public\_attributes

```
const pkcs11_attr_model pkcs11_cert_x509public_attributes[]
```

CKO\_CERTIFICATE (Type: CKC\_X\_509) - X509 Public Key Certificate Model

### 18.13.5.6 pkcs11\_cert\_x509public\_attributes\_count

```
const CK_ULONG pkcs11_cert_x509public_attributes_count = sizeof( pkcs11_cert_x509public_attributes ) / sizeof( pkcs11_cert_x509public_attributes [0])
```

### 18.13.5.7 pkcs11\_key\_ec\_private\_attributes

```
const pkcs11_attrib_model pkcs11_key_ec_private_attributes[]
```

#### Initial value:

```
= {  
    { 0x00000180UL , pkcs11_key_get_ec_params },  
    { 0x00000181UL , pkcs11_key_get_ec_point  },  
}
```

CKO\_PRIVATE\_KEY (Type: CKK\_EC) - EC/ECDSA Public Key Object Model

### 18.13.5.8 pkcs11\_key\_ec\_public\_attributes

```
const pkcs11_attrib_model pkcs11_key_ec_public_attributes[]
```

#### Initial value:

```
= {  
    { 0x00000180UL , pkcs11_key_get_ec_params },  
    { 0x00000181UL , pkcs11_key_get_ec_point  },  
}
```

CKO\_PUBLIC\_KEY (Type: CKK\_EC) - EC/ECDSA Public Key Object Model

### 18.13.5.9 pkcs11\_key\_private\_attributes

```
const pkcs11_attrib_model pkcs11_key_private_attributes[]
```

CKO\_PRIVATE\_KEY - Private Key Object Base Model

### 18.13.5.10 pkcs11\_key\_private\_attributes\_count

```
const CK_ULONG pkcs11_key_private_attributes_count = sizeof( pkcs11_key_private_attributes ) /  
sizeof( pkcs11_key_private_attributes [0])
```

### 18.13.5.11 pkcs11\_key\_public\_attributes

```
const pkcs11_attrib_model pkcs11_key_public_attributes[]
```

CKO\_PUBLIC\_KEY - Public Key Object Model

### 18.13.5.12 pkcs11\_key\_public\_attributes\_count

```
const CK_ULONG pkcs11_key_public_attributes_count = sizeof( pkcs11_key_public_attributes ) /  
sizeof( pkcs11_key_public_attributes [0])
```

**18.13.5.13 pkcs11\_key\_rsa\_private\_attributes**

```
const pkcs11\_attrib\_model pkcs11_key_rsa_private_attributes[]
```

**Initial value:**

```
= {
    { 0x00000120UL ,      0 ,      },
    { 0x00000122UL ,    0 ,      },
    { 0x00000123UL ,    0 ,      },
    { 0x00000124UL ,      0 ,      },
    { 0x00000125UL ,      0 ,      },
    { 0x00000126UL ,      0 ,      },
    { 0x00000127UL ,      0 ,      },
    { 0x00000128UL ,      0 ,      },
}
```

CKO\_PRIVATE\_KEY (Type: CKK\_RSA) - RSA Private Key Object Model

**18.13.5.14 pkcs11\_key\_secret\_attributes**

```
const pkcs11\_attrib\_model pkcs11_key_secret_attributes[]
```

CKO\_SECRET\_KEY - Secret Key Object Base Model

**18.13.5.15 pkcs11\_key\_secret\_attributes\_count**

```
const CK\_ULONG pkcs11_key_secret_attributes_count = sizeof( pkcs11\_key\_secret\_attributes ) /
sizeof( pkcs11\_key\_secret\_attributes [0])
```

**18.13.5.16 pkcs11\_lib\_description**

```
const char pkcs11_lib_description[] = "Cryptoauthlib PKCS11 Interface"
```

**18.13.5.17 pkcs11\_lib\_manufacturer\_id**

```
const char pkcs11_lib_manufacturer_id[] = "Microchip Technology Inc"
```

**18.13.5.18 pkcs11\_object\_cache**

```
pkcs11\_object\_cache\_t pkcs11_object_cache[PKCS11_MAX_OBJECTS_ALLOWED]
```

### 18.13.5.19 pkcs11\_object\_monotonic\_attributes

```
const pkcs11_attrib_model pkcs11_object_monotonic_attributes[]
```

**Initial value:**

```
= {  
    { 0x00000000UL ,      pkcs11_object_get_class  
      },  
    { 0x00000300UL , pkcs11_object_get_type      },  
    { 0x00000301UL ,      pkcs11_attrib_false      },  
    { 0x00000302UL ,      pkcs11_attrib_false      },  
    { 0x00000011UL ,      0                        },  
}
```

```
CKA_CLASS == CKO_HW_FEATURE_TYPE CKA_HW_FEATURE_TYPE == CKH_MONOTONIC_COUNTER
```

### 18.13.5.20 pkcs11\_object\_monotonic\_attributes\_count

```
const CK_ULONG pkcs11_object_monotonic_attributes_count = sizeof( pkcs11_object_monotonic_attributes  
 ) / sizeof( pkcs11_object_monotonic_attributes [0])
```

## 18.14 TNG API (tng\_)

These methods provide some convenience functions (mostly around certificates) for TNG devices, which currently include ATECC608A-MAHTN-T.

### Functions

- const [atcacert\\_def\\_t \\* tng\\_map\\_get\\_device\\_cert\\_def](#) (int index)  
*Helper function to iterate through all trust cert definitions.*
- [ATCA\\_STATUS tng\\_get\\_device\\_cert\\_def](#) (const [atcacert\\_def\\_t](#) \*\*cert\_def)  
*Get the TNG device certificate definition.*
- [ATCA\\_STATUS tng\\_get\\_device\\_pubkey](#) (uint8\_t \*public\_key)  
*Uses GenKey command to calculate the public key from the primary device public key.*
- const [atcacert\\_def\\_t g\\_tfltls\\_cert\\_def\\_4\\_device](#)
- int [tng\\_atcacert\\_max\\_device\\_cert\\_size](#) (size\_t \*max\_cert\_size)  
*Return the maximum possible certificate size in bytes for a TNG device certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.*
- int [tng\\_atcacert\\_read\\_device\\_cert](#) (uint8\_t \*cert, size\_t \*cert\_size, const uint8\_t \*signer\_cert)  
*Reads the device certificate for a TNG device.*
- int [tng\\_atcacert\\_device\\_public\\_key](#) (uint8\_t \*public\_key, uint8\_t \*cert)  
*Reads the device public key.*
- int [tng\\_atcacert\\_max\\_signer\\_cert\\_size](#) (size\_t \*max\_cert\_size)  
*Return the maximum possible certificate size in bytes for a TNG signer certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.*
- int [tng\\_atcacert\\_read\\_signer\\_cert](#) (uint8\_t \*cert, size\_t \*cert\_size)  
*Reads the signer certificate for a TNG device.*
- int [tng\\_atcacert\\_signer\\_public\\_key](#) (uint8\_t \*public\_key, uint8\_t \*cert)  
*Reads the signer public key.*
- int [tng\\_atcacert\\_root\\_cert\\_size](#) (size\_t \*cert\_size)  
*Get the size of the TNG root cert.*
- int [tng\\_atcacert\\_root\\_cert](#) (uint8\_t \*cert, size\_t \*cert\_size)  
*Get the TNG root cert.*
- int [tng\\_atcacert\\_root\\_public\\_key](#) (uint8\_t \*public\_key)  
*Gets the root public key.*
- const uint8\_t [g\\_cryptoauth\\_root\\_ca\\_002\\_cert](#) []
- const size\_t [g\\_cryptoauth\\_root\\_ca\\_002\\_cert\\_size](#)
- #define [CRYPTOAUTH\\_ROOT\\_CA\\_002\\_PUBLIC\\_KEY\\_OFFSET](#) 266
- const [atcacert\\_def\\_t g\\_tnglora\\_cert\\_def\\_1\\_signer](#)
- const [atcacert\\_def\\_t g\\_tnglora\\_cert\\_def\\_2\\_device](#)
- const [atcacert\\_def\\_t g\\_tnglora\\_cert\\_def\\_4\\_device](#)
- #define [TNGLORA\\_CERT\\_TEMPLATE\\_4\\_DEVICE\\_SIZE](#) 552
- [ATCA\\_DLL](#) const [atcacert\\_def\\_t g\\_tngtls\\_cert\\_def\\_1\\_signer](#)
- #define [TNGTLS\\_CERT\\_TEMPLATE\\_1\\_SIGNER\\_SIZE](#) 520
- const [atcacert\\_def\\_t g\\_tngtls\\_cert\\_def\\_2\\_device](#)
- #define [TNGTLS\\_CERT\\_TEMPLATE\\_2\\_DEVICE\\_SIZE](#) 505
- #define [TNGTLS\\_CERT\\_ELEMENTS\\_2\\_DEVICE\\_COUNT](#) 2
- const [atcacert\\_def\\_t g\\_tngtls\\_cert\\_def\\_3\\_device](#)
- #define [TNGTLS\\_CERT\\_TEMPLATE\\_3\\_DEVICE\\_SIZE](#) 546

### 18.14.1 Detailed Description

These methods provide some convenience functions (mostly around certificates) for TNG devices, which currently include ATECC608A-MAHTN-T.

### 18.14.2 Macro Definition Documentation

#### 18.14.2.1 CRYPTOAUTH\_ROOT\_CA\_002\_PUBLIC\_KEY\_OFFSET

```
#define CRYPTOAUTH_ROOT_CA_002_PUBLIC_KEY_OFFSET 266
```

#### 18.14.2.2 TNGLORA\_CERT\_TEMPLATE\_4\_DEVICE\_SIZE

```
#define TNGLORA_CERT_TEMPLATE_4_DEVICE_SIZE 552
```

#### 18.14.2.3 TNGTLS\_CERT\_ELEMENTS\_2\_DEVICE\_COUNT

```
#define TNGTLS_CERT_ELEMENTS_2_DEVICE_COUNT 2
```

#### 18.14.2.4 TNGTLS\_CERT\_TEMPLATE\_1\_SIGNER\_SIZE

```
#define TNGTLS_CERT_TEMPLATE_1_SIGNER_SIZE 520
```

#### 18.14.2.5 TNGTLS\_CERT\_TEMPLATE\_2\_DEVICE\_SIZE

```
#define TNGTLS_CERT_TEMPLATE_2_DEVICE_SIZE 505
```

#### 18.14.2.6 TNGTLS\_CERT\_TEMPLATE\_3\_DEVICE\_SIZE

```
#define TNGTLS_CERT_TEMPLATE_3_DEVICE_SIZE 546
```

### 18.14.3 Function Documentation

#### 18.14.3.1 tng\_atcacert\_device\_public\_key()

```
int tng_atcacert_device_public_key (  
    uint8_t * public_key,  
    uint8_t * cert )
```

Reads the device public key.

## Parameters

out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve.
in	<i>cert</i>	If supplied, the device public key is used from this certificate. If set to NULL, the device public key is read from the device.

## Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.14.3.2 tng\_atcacert\_max\_device\_cert\_size()**

```
int tng_atcacert_max_device_cert_size (
    size_t * max_cert_size )
```

Return the maximum possible certificate size in bytes for a TNG device certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.

## Parameters

out	<i>max_cert_size</i>	Maximum certificate size will be returned here in bytes.
-----	----------------------	--

## Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.14.3.3 tng\_atcacert\_max\_signer\_cert\_size()**

```
int tng_atcacert_max_signer_cert_size (
    size_t * max_cert_size )
```

Return the maximum possible certificate size in bytes for a TNG signer certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.

## Parameters

out	<i>max_cert_size</i>	Maximum certificate size will be returned here in bytes.
-----	----------------------	--

## Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

## 18.14 TNG API (tng\_)

---

### 18.14.3.4 tng\_atcacert\_read\_device\_cert()

```
int tng_atcacert_read_device_cert (
    uint8_t * cert,
    size_t * cert_size,
    const uint8_t * signer_cert )
```

Reads the device certificate for a TNG device.

#### Parameters

out	<i>cert</i>	Buffer to received the certificate (DER format).
in, out	<i>cert_size</i>	As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in cert in bytes.
in	<i>signer_cert</i>	If supplied, the signer public key is used from this certificate. If set to NULL, the signer public key is read from the device.

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.14.3.5 tng\_atcacert\_read\_signer\_cert()

```
int tng_atcacert_read_signer_cert (
    uint8_t * cert,
    size_t * cert_size )
```

Reads the signer certificate for a TNG device.

#### Parameters

out	<i>cert</i>	Buffer to received the certificate (DER format).
in, out	<i>cert_size</i>	As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in cert in bytes.

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.14.3.6 tng\_atcacert\_root\_cert()

```
int tng_atcacert_root_cert (
    uint8_t * cert,
    size_t * cert_size )
```

Get the TNG root cert.



**Parameters**

out	<i>cert</i>	Buffer to received the certificate (DER format).
in, out	<i>cert_size</i>	As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in cert in bytes.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.14.3.7 tng\_atcacert\_root\_cert\_size()**

```
int tng_atcacert_root_cert_size (
    size_t * cert_size )
```

Get the size of the TNG root cert.

**Parameters**

out	<i>cert_size</i>	Certificate size will be returned here in bytes.
-----	------------------	--

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**18.14.3.8 tng\_atcacert\_root\_public\_key()**

```
int tng_atcacert_root_public_key (
    uint8_t * public_key )
```

Gets the root public key.

**Parameters**

out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve.
-----	-------------------	--

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

## 18.14 TNG API (tng\_)

---

### 18.14.3.9 tng\_atcacert\_signer\_public\_key()

```
int tng_atcacert_signer_public_key (
    uint8_t * public_key,
    uint8_t * cert )
```

Reads the signer public key.

#### Parameters

out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve.
in	<i>cert</i>	If supplied, the signer public key is used from this certificate. If set to NULL, the signer public key is read from the device.

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 18.14.3.10 tng\_get\_device\_cert\_def()

```
ATCA_STATUS tng_get_device_cert_def (
    const atcacert_def_t ** cert_def )
```

Get the TNG device certificate definition.

#### Parameters

out	<i>cert_def</i>	TNG device certificate definition is returned here.
-----	-----------------	---

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 18.14.3.11 tng\_get\_device\_pubkey()

```
ATCA_STATUS tng_get_device_pubkey (
    uint8_t * public_key )
```

Uses GenKey command to calculate the public key from the primary device public key.

#### Parameters

out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve.
-----	-------------------	--

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**18.14.3.12 tng\_map\_get\_device\_cert\_def()**

```
const atcacert_def_t* tng_map_get_device_cert_def (
    int index )
```

Helper function to iterate through all trust cert definitions.

**Parameters**

in	<i>index</i>	Map index
----	--------------	-----------

**Returns**

non-null value if success, otherwise NULL

**18.14.4 Variable Documentation****18.14.4.1 g\_cryptoauth\_root\_ca\_002\_cert**

```
const uint8_t g_cryptoauth_root_ca_002_cert[]
```

**18.14.4.2 g\_cryptoauth\_root\_ca\_002\_cert\_size**

```
const size_t g_cryptoauth_root_ca_002_cert_size
```

**18.14.4.3 g\_tflxtls\_cert\_def\_4\_device**

```
const atcacert_def_t g_tflxtls_cert_def_4_device
```

**18.14.4.4 g\_tnqlora\_cert\_def\_1\_signer**

```
const atcacert_def_t g_tnqlora_cert_def_1_signer
```

### 18.14.4.5 g\_tnglora\_cert\_def\_2\_device

```
const atcacert_def_t g_tnglora_cert_def_2_device
```

### 18.14.4.6 g\_tnglora\_cert\_def\_4\_device

```
const atcacert_def_t g_tnglora_cert_def_4_device
```

### 18.14.4.7 g\_tngtls\_cert\_def\_1\_signer

```
ATCA_DLL const atcacert_def_t g_tngtls_cert_def_1_signer
```

### 18.14.4.8 g\_tngtls\_cert\_def\_2\_device

```
const atcacert_def_t g_tngtls_cert_def_2_device
```

### 18.14.4.9 g\_tngtls\_cert\_def\_3\_device

```
const atcacert_def_t g_tngtls_cert_def_3_device
```



## Chapter 19

# Data Structure Documentation

### 19.1 `_atecc508a_config` Struct Reference

```
#include <atca_device.h>
```

#### Data Fields

- `uint32_t` [SN03](#)
- `uint32_t` [RevNum](#)
- `uint32_t` [SN47](#)
- `uint8_t` [SN8](#)
- `uint8_t` [Reserved0](#)
- `uint8_t` [I2C\\_Enable](#)
- `uint8_t` [Reserved1](#)
- `uint8_t` [I2C\\_Address](#)
- `uint8_t` [Reserved2](#)
- `uint8_t` [OTPmode](#)
- `uint8_t` [ChipMode](#)
- `uint16_t` [SlotConfig](#) [16]
- `uint8_t` [Counter0](#) [8]
- `uint8_t` [Counter1](#) [8]
- `uint8_t` [LastKeyUse](#) [16]
- `uint8_t` [UserExtra](#)
- `uint8_t` [Selector](#)
- `uint8_t` [LockValue](#)
- `uint8_t` [LockConfig](#)
- `uint16_t` [SlotLocked](#)
- `uint16_t` [RFU](#)
- `uint32_t` [X509format](#)
- `uint16_t` [KeyConfig](#) [16]

#### 19.1.1 Field Documentation

**19.1.1.1 ChipMode**

```
uint8_t ChipMode
```

**19.1.1.2 Counter0**

```
uint8_t Counter0[8]
```

**19.1.1.3 Counter1**

```
uint8_t Counter1[8]
```

**19.1.1.4 I2C\_Address**

```
uint8_t I2C_Address
```

**19.1.1.5 I2C\_Enable**

```
uint8_t I2C_Enable
```

**19.1.1.6 KeyConfig**

```
uint16_t KeyConfig[16]
```

**19.1.1.7 LastKeyUse**

```
uint8_t LastKeyUse[16]
```

**19.1.1.8 LockConfig**

```
uint8_t LockConfig
```

### 19.1.1.9 LockValue

uint8\_t LockValue

### 19.1.1.10 OTPmode

uint8\_t OTPmode

### 19.1.1.11 Reserved0

uint8\_t Reserved0

### 19.1.1.12 Reserved1

uint8\_t Reserved1

### 19.1.1.13 Reserved2

uint8\_t Reserved2

### 19.1.1.14 RevNum

uint32\_t RevNum

### 19.1.1.15 RFU

uint16\_t RFU

### 19.1.1.16 Selector

uint8\_t Selector



**19.1.1.17 SlotConfig**

```
uint16_t SlotConfig[16]
```

**19.1.1.18 SlotLocked**

```
uint16_t SlotLocked
```

**19.1.1.19 SN03**

```
uint32_t SN03
```

**19.1.1.20 SN47**

```
uint32_t SN47
```

**19.1.1.21 SN8**

```
uint8_t SN8
```

**19.1.1.22 UserExtra**

```
uint8_t UserExtra
```

**19.1.1.23 X509format**

```
uint32_t X509format
```

**19.2 \_atecc608\_config Struct Reference**

```
#include <atca_device.h>
```

### Data Fields

- uint32\_t [SN03](#)
- uint32\_t [RevNum](#)
- uint32\_t [SN47](#)
- uint8\_t [SN8](#)
- uint8\_t [AES\\_Enable](#)
- uint8\_t [I2C\\_Enable](#)
- uint8\_t [Reserved1](#)
- uint8\_t [I2C\\_Address](#)
- uint8\_t [Reserved2](#)
- uint8\_t [CountMatch](#)
- uint8\_t [ChipMode](#)
- uint16\_t [SlotConfig](#) [16]
- uint8\_t [Counter0](#) [8]
- uint8\_t [Counter1](#) [8]
- uint8\_t [UseLock](#)
- uint8\_t [VolatileKeyPermission](#)
- uint16\_t [SecureBoot](#)
- uint8\_t [KdfIvLoc](#)
- uint16\_t [KdfIvStr](#)
- uint8\_t [Reserved3](#) [9]
- uint8\_t [UserExtra](#)
- uint8\_t [UserExtraAdd](#)
- uint8\_t [LockValue](#)
- uint8\_t [LockConfig](#)
- uint16\_t [SlotLocked](#)
- uint16\_t [ChipOptions](#)
- uint32\_t [X509format](#)
- uint16\_t [KeyConfig](#) [16]

### 19.2.1 Field Documentation

#### 19.2.1.1 AES\_Enable

uint8\_t AES\_Enable

#### 19.2.1.2 ChipMode

uint8\_t ChipMode

### 19.2.1.3 ChipOptions

uint16\_t ChipOptions

### 19.2.1.4 Counter0

uint8\_t Counter0[8]

### 19.2.1.5 Counter1

uint8\_t Counter1[8]

### 19.2.1.6 CountMatch

uint8\_t CountMatch

### 19.2.1.7 I2C\_Address

uint8\_t I2C\_Address

### 19.2.1.8 I2C\_Enable

uint8\_t I2C\_Enable

### 19.2.1.9 KdfIvLoc

uint8\_t KdfIvLoc

### 19.2.1.10 KdfIvStr

uint16\_t KdfIvStr

### 19.2.1.11 KeyConfig

uint16\_t KeyConfig[16]

### 19.2.1.12 LockConfig

uint8\_t LockConfig

### 19.2.1.13 LockValue

uint8\_t LockValue

### 19.2.1.14 Reserved1

uint8\_t Reserved1

### 19.2.1.15 Reserved2

uint8\_t Reserved2

### 19.2.1.16 Reserved3

uint8\_t Reserved3[9]

### 19.2.1.17 RevNum

uint32\_t RevNum

### 19.2.1.18 SecureBoot

uint16\_t SecureBoot

**19.2.1.19 SlotConfig**

```
uint16_t SlotConfig[16]
```

**19.2.1.20 SlotLocked**

```
uint16_t SlotLocked
```

**19.2.1.21 SN03**

```
uint32_t SN03
```

**19.2.1.22 SN47**

```
uint32_t SN47
```

**19.2.1.23 SN8**

```
uint8_t SN8
```

**19.2.1.24 UseLock**

```
uint8_t UseLock
```

**19.2.1.25 UserExtra**

```
uint8_t UserExtra
```

**19.2.1.26 UserExtraAdd**

```
uint8_t UserExtraAdd
```

### 19.2.1.27 VolatileKeyPermission

uint8\_t VolatileKeyPermission

### 19.2.1.28 X509format

uint32\_t X509format

## 19.3 \_atsha204a\_config Struct Reference

```
#include <atca_device.h>
```

### Data Fields

- uint32\_t [SN03](#)
- uint32\_t [RevNum](#)
- uint32\_t [SN47](#)
- uint8\_t [SN8](#)
- uint8\_t [Reserved0](#)
- uint8\_t [I2C\\_Enable](#)
- uint8\_t [Reserved1](#)
- uint8\_t [I2C\\_Address](#)
- uint8\_t [Reserved2](#)
- uint8\_t [OTPmode](#)
- uint8\_t [ChipMode](#)
- uint16\_t [SlotConfig](#) [16]
- uint16\_t [Counter](#) [8]
- uint8\_t [LastKeyUse](#) [16]
- uint8\_t [UserExtra](#)
- uint8\_t [Selector](#)
- uint8\_t [LockValue](#)
- uint8\_t [LockConfig](#)

### 19.3.1 Field Documentation

#### 19.3.1.1 ChipMode

uint8\_t ChipMode

**19.3.1.2 Counter**

```
uint16_t Counter[8]
```

**19.3.1.3 I2C\_Address**

```
uint8_t I2C_Address
```

**19.3.1.4 I2C\_Enable**

```
uint8_t I2C_Enable
```

**19.3.1.5 LastKeyUse**

```
uint8_t LastKeyUse[16]
```

**19.3.1.6 LockConfig**

```
uint8_t LockConfig
```

**19.3.1.7 LockValue**

```
uint8_t LockValue
```

**19.3.1.8 OTPmode**

```
uint8_t OTPmode
```

**19.3.1.9 Reserved0**

```
uint8_t Reserved0
```

### 19.3.1.10 Reserved1

uint8\_t Reserved1

### 19.3.1.11 Reserved2

uint8\_t Reserved2

### 19.3.1.12 RevNum

uint32\_t RevNum

### 19.3.1.13 Selector

uint8\_t Selector

### 19.3.1.14 SlotConfig

uint16\_t SlotConfig[16]

### 19.3.1.15 SN03

uint32\_t SN03

### 19.3.1.16 SN47

uint32\_t SN47

### 19.3.1.17 SN8

uint8\_t SN8



### 19.3.1.18 UserExtra

`uint8_t UserExtra`

## 19.4 \_pkcs11\_mech\_table\_e Struct Reference

### Data Fields

- [CK\\_MECHANISM\\_TYPE](#) type
- [CK\\_MECHANISM\\_INFO](#) info

### 19.4.1 Field Documentation

#### 19.4.1.1 info

[CK\\_MECHANISM\\_INFO](#) info

#### 19.4.1.2 type

[CK\\_MECHANISM\\_TYPE](#) type

## 19.5 \_pkcs11\_attr\_model Struct Reference

```
#include <pkcs11_attr.h>
```

### Data Fields

- const [CK\\_ATTRIBUTE\\_TYPE](#) type
- const [attrib\\_f](#) func

### 19.5.1 Field Documentation

#### 19.5.1.1 func

const [attrib\\_f](#) func

### 19.5.1.2 type

const [CK\\_ATTRIBUTE\\_TYPE](#) type

## 19.6 \_pkcs11\_lib\_ctx Struct Reference

```
#include <pkcs11_init.h>
```

### Data Fields

- [CK\\_BBOOL](#) initialized
- CK\_CREATEMUTEX [create\\_mutex](#)
- CK\_DESTROYMUTEX [destroy\\_mutex](#)
- CK\_LOCKMUTEX [lock\\_mutex](#)
- CK\_UNLOCKMUTEX [unlock\\_mutex](#)
- [CK\\_VOID\\_PTR](#) mutex
- [CK\\_VOID\\_PTR](#) slots
- [CK\\_ULONG](#) slot\_cnt
- [CK\\_CHAR](#) config\_path [200]

### 19.6.1 Detailed Description

Library Context

### 19.6.2 Field Documentation

#### 19.6.2.1 config\_path

[CK\\_CHAR](#) config\_path[200]

#### 19.6.2.2 create\_mutex

CK\_CREATEMUTEX create\_mutex

#### 19.6.2.3 destroy\_mutex

CK\_DESTROYMUTEX destroy\_mutex

#### 19.6.2.4 initialized

`CK_BBOOL` initialized

#### 19.6.2.5 lock\_mutex

`CK_LOCKMUTEX` lock\_mutex

#### 19.6.2.6 mutex

`CK_VOID_PTR` mutex

#### 19.6.2.7 slot\_cnt

`CK_ULONG` slot\_cnt

#### 19.6.2.8 slots

`CK_VOID_PTR` slots

#### 19.6.2.9 unlock\_mutex

`CK_UNLOCKMUTEX` unlock\_mutex

### 19.7 \_pkcs11\_object Struct Reference

```
#include <pkcs11_object.h>
```

### Data Fields

- [CK\\_OBJECT\\_CLASS](#) [class\\_id](#)
- [CK\\_ULONG](#) [class\\_type](#)
- [pkcs11\\_attr\\_model](#) [const](#) \* [attributes](#)
- [CK\\_ULONG](#) [count](#)
- [CK\\_ULONG](#) [size](#)
- [uint16\\_t](#) [slot](#)
- [CK\\_FLAGS](#) [flags](#)
- [CK\\_UTF8CHAR](#) [name](#) [PKCS11\_MAX\_LABEL\_SIZE+1]
- [CK\\_VOID\\_PTR](#) [config](#)
- [CK\\_VOID\\_PTR](#) [data](#)
- [ta\\_element\\_attributes\\_t](#) [handle\\_info](#)

### 19.7.1 Field Documentation

#### 19.7.1.1 attributes

[pkcs11\\_attr\\_model](#) [const](#)\* [attributes](#)

List of attribute models this object possesses

#### 19.7.1.2 class\_id

[CK\\_OBJECT\\_CLASS](#) [class\\_id](#)

The Class Identifier

#### 19.7.1.3 class\_type

[CK\\_ULONG](#) [class\\_type](#)

The Class Type

#### 19.7.1.4 config

[CK\\_VOID\\_PTR](#) [config](#)

#### 19.7.1.5 count

[CK\\_ULONG](#) [count](#)

Count of attribute models

#### 19.7.1.6 data

`CK_VOID_PTR` data

#### 19.7.1.7 flags

`CK_FLAGS` flags

#### 19.7.1.8 handle\_info

`ta_element_attributes_t` handle\_info

#### 19.7.1.9 name

`CK_UTF8CHAR` name[PKCS11\_MAX\_LABEL\_SIZE+1]

#### 19.7.1.10 size

`CK_ULONG` size

#### 19.7.1.11 slot

`uint16_t` slot

## 19.8 \_pkcs11\_object\_cache\_t Struct Reference

```
#include <pkcs11_object.h>
```

### Data Fields

- `CK_OBJECT_HANDLE` handle
- `pkcs11_object_ptr` object

### 19.8.1 Field Documentation

#### 19.8.1.1 handle

`CK_OBJECT_HANDLE` handle

Arbitrary (but unique) non-null identifier for an object

#### 19.8.1.2 object

`pkcs11_object_ptr` object

The actual object

## 19.9 \_pkcs11\_session\_ctx Struct Reference

```
#include <pkcs11_session.h>
```

### Data Fields

- `CK_BBOOL` initialized
- `pkcs11_slot_ctx_ptr` slot
- `CK_SESSION_HANDLE` handle
- `CK_STATE` state
- `CK_ULONG` error
- `CK_ATTRIBUTE_PTR` attrib\_list
- `CK_ULONG` attrib\_count
- `CK_ULONG` object\_index
- `CK_ULONG` object\_count
- `CK_OBJECT_HANDLE` active\_object
- `CK_BBOOL` logged\_in
- `CK_BYTE` read\_key [32]

### 19.9.1 Detailed Description

Session Context

### 19.9.2 Field Documentation

**19.9.2.1 active\_object**

`CK_OBJECT_HANDLE` active\_object

**19.9.2.2 attrib\_count**

`CK_ULONG` attrib\_count

**19.9.2.3 attrib\_list**

`CK_ATTRIBUTE_PTR` attrib\_list

**19.9.2.4 error**

`CK_ULONG` error

**19.9.2.5 handle**

`CK_SESSION_HANDLE` handle

**19.9.2.6 initialized**

`CK_BBOOL` initialized

**19.9.2.7 logged\_in**

`CK_BBOOL` logged\_in

**19.9.2.8 object\_count**

`CK_ULONG` object\_count

## 19.10 \_pkcs11\_slot\_ctx Struct Reference

---

### 19.9.2.9 object\_index

`CK_ULONG` object\_index

### 19.9.2.10 read\_key

`CK_BYTE` read\_key[32]

Accepted through C\_Login as the user pin

### 19.9.2.11 slot

`pkcs11_slot_ctx_ptr` slot

### 19.9.2.12 state

`CK_STATE` state

## 19.10 \_pkcs11\_slot\_ctx Struct Reference

```
#include <pkcs11_slot.h>
```

### Data Fields

- `CK_BBOOL` initialized
- `CK_SLOT_ID` slot\_id
- `ATCADevice` device\_ctx
- `ATCAIfaceCfg` interface\_config
- `CK_SESSION_HANDLE` session
- `atecc608_config_t` cfg\_zone
- `CK_FLAGS` flags
- `uint16_t` user\_pin\_handle
- `uint16_t` so\_pin\_handle
- `CK_UTF8CHAR` label [PKCS11\_MAX\_LABEL\_SIZE+1]

### 19.10.1 Detailed Description

Slot Context



## 19.10.2 Field Documentation

### 19.10.2.1 `cfg_zone`

`atecc608_config_t` `cfg_zone`

### 19.10.2.2 `device_ctx`

`ATCADevice` `device_ctx`

### 19.10.2.3 `flags`

`CK_FLAGS` `flags`

### 19.10.2.4 `initialized`

`CK_BBOOL` `initialized`

### 19.10.2.5 `interface_config`

`ATCAIfaceCfg` `interface_config`

### 19.10.2.6 `label`

`CK_UTF8CHAR` `label` `[PKCS11_MAX_LABEL_SIZE+1]`

### 19.10.2.7 `session`

`CK_SESSION_HANDLE` `session`

### 19.10.2.8 slot\_id

[CK\\_SLOT\\_ID](#) slot\_id

### 19.10.2.9 so\_pin\_handle

uint16\_t so\_pin\_handle

### 19.10.2.10 user\_pin\_handle

uint16\_t user\_pin\_handle

## 19.11 atca\_aes\_cbc\_ctx Struct Reference

```
#include <atca_crypto_hw_aes.h>
```

### Data Fields

- [ATCADevice device](#)  
*Device Context Pointer.*
- uint16\_t [key\\_id](#)  
*Key location. Can either be a slot number or ATCA\_TEMPKEY\_KEYID for TempKey.*
- uint8\_t [key\\_block](#)  
*Index of the 16-byte block to use within the key location for the actual key.*
- uint8\_t [ciphertext](#) [[ATCA\\_AES128\\_BLOCK\\_SIZE](#)]  
*Ciphertext from last operation.*

### 19.11.1 Field Documentation

#### 19.11.1.1 ciphertext

uint8\_t ciphertext [[ATCA\\_AES128\\_BLOCK\\_SIZE](#)]

Ciphertext from last operation.

### 19.11.1.2 device

[ATCADevice](#) device

Device Context Pointer.

### 19.11.1.3 key\_block

uint8\_t key\_block

Index of the 16-byte block to use within the key location for the actual key.

### 19.11.1.4 key\_id

uint16\_t key\_id

Key location. Can either be a slot number or ATCA\_TEMPKEY\_KEYID for TempKey.

## 19.12 atca\_aes\_cmac\_ctx Struct Reference

```
#include <atca_crypto_hw_aes.h>
```

### Data Fields

- [atca\\_aes\\_cbc\\_ctx\\_t cbc\\_ctx](#)  
*CBC context.*
- uint32\_t [block\\_size](#)  
*Number of bytes in current block.*
- uint8\_t [block](#) [[ATCA\\_AES128\\_BLOCK\\_SIZE](#)]  
*Unprocessed message storage.*

### 19.12.1 Field Documentation

#### 19.12.1.1 block

uint8\_t block [[ATCA\\_AES128\\_BLOCK\\_SIZE](#)]

Unprocessed message storage.

### 19.12.1.2 block\_size

uint32\_t block\_size

Number of bytes in current block.

### 19.12.1.3 cbc\_ctx

atca\_aes\_cbc\_ctx\_t cbc\_ctx

CBC context.

## 19.13 atca\_aes\_ctr\_ctx Struct Reference

```
#include <atca_crypto_hw_aes.h>
```

### Data Fields

- [ATCADevice device](#)  
*Device Context Pointer.*
- uint16\_t [key\\_id](#)  
*Key location. Can either be a slot number or ATCA\_TEMPKEY\_KEYID for TempKey.*
- uint8\_t [key\\_block](#)  
*Index of the 16-byte block to use within the key location for the actual key.*
- uint8\_t [cb](#) [[ATCA\\_AES128\\_BLOCK\\_SIZE](#)]  
*Counter block, comprises of nonce + count value (16 bytes).*
- uint8\_t [counter\\_size](#)  
*Size of counter in the initialization vector.*

### 19.13.1 Field Documentation

#### 19.13.1.1 cb

uint8\_t cb [[ATCA\\_AES128\\_BLOCK\\_SIZE](#)]

Counter block, comprises of nonce + count value (16 bytes).

#### 19.13.1.2 counter\_size

```
uint8_t counter_size
```

Size of counter in the initialization vector.

#### 19.13.1.3 device

```
ATCADevice device
```

Device Context Pointer.

#### 19.13.1.4 key\_block

```
uint8_t key_block
```

Index of the 16-byte block to use within the key location for the actual key.

#### 19.13.1.5 key\_id

```
uint16_t key_id
```

Key location. Can either be a slot number or ATCA\_TEMPKEY\_KEYID for TempKey.

### 19.14 atca\_aes\_gcm\_ctx Struct Reference

```
#include <calib_aes_gcm.h>
```

## Data Fields

- uint16\_t [key\\_id](#)  
*Key location. Can either be a slot number or ATCA\_TEMPKEY\_KEYID for TempKey.*
- uint8\_t [key\\_block](#)  
*Index of the 16-byte block to use within the key location for the actual key.*
- uint8\_t [cb](#) [AES\_DATA\_SIZE]  
*Counter block, comprises of nonce + count value (16 bytes).*
- uint32\_t [data\\_size](#)  
*Size of the data being encrypted/decrypted in bytes.*
- uint32\_t [aad\\_size](#)  
*Size of the additional authenticated data in bytes.*
- uint8\_t [h](#) [AES\_DATA\_SIZE]  
*Subkey for ghash functions in GCM.*
- uint8\_t [j0](#) [AES\_DATA\_SIZE]  
*Precounter block generated from IV.*
- uint8\_t [y](#) [AES\_DATA\_SIZE]  
*Current GHASH output.*
- uint8\_t [partial\\_aad](#) [AES\_DATA\_SIZE]  
*Partial blocks of data waiting to be processed.*
- uint32\_t [partial\\_aad\\_size](#)  
*Amount of data in the partial block buffer.*
- uint8\_t [enc\\_cb](#) [AES\_DATA\_SIZE]  
*Last encrypted counter block.*
- uint8\_t [ciphertext\\_block](#) [AES\_DATA\_SIZE]  
*Last ciphertext block.*

### 19.14.1 Detailed Description

Context structure for AES GCM operations.

### 19.14.2 Field Documentation

#### 19.14.2.1 aad\_size

```
uint32_t aad_size
```

Size of the additional authenticated data in bytes.

#### 19.14.2.2 cb

```
uint8_t cb[AES_DATA_SIZE]
```

Counter block, comprises of nonce + count value (16 bytes).

#### 19.14.2.3 ciphertext\_block

```
uint8_t ciphertext_block[AES_DATA_SIZE]
```

Last ciphertext block.

#### 19.14.2.4 data\_size

```
uint32_t data_size
```

Size of the data being encrypted/decrypted in bytes.

#### 19.14.2.5 enc\_cb

```
uint8_t enc_cb[AES_DATA_SIZE]
```

Last encrypted counter block.

#### 19.14.2.6 h

```
uint8_t h[AES_DATA_SIZE]
```

Subkey for ghash functions in GCM.

#### 19.14.2.7 j0

```
uint8_t j0[AES_DATA_SIZE]
```

Precounter block generated from IV.

#### 19.14.2.8 key\_block

```
uint8_t key_block
```

Index of the 16-byte block to use within the key location for the actual key.

### 19.14.2.9 key\_id

uint16\_t key\_id

Key location. Can either be a slot number or ATCA\_TEMPKEY\_KEYID for TempKey.

### 19.14.2.10 partial\_aad

uint8\_t partial\_aad[AES\_DATA\_SIZE]

Partial blocks of data waiting to be processed.

### 19.14.2.11 partial\_aad\_size

uint32\_t partial\_aad\_size

Amount of data in the partial block buffer.

### 19.14.2.12 y

uint8\_t y[AES\_DATA\_SIZE]

Current GHASH output.

## 19.15 atca\_check\_mac\_in\_out Struct Reference

Input/output parameters for function [atcah\\_check\\_mac\(\)](#).

```
#include <atca_host.h>
```

### Data Fields

- uint8\_t [mode](#)  
*[in] CheckMac command Mode*
- uint16\_t [key\\_id](#)  
*[in] CheckMac command KeyID*
- const uint8\_t \* [sn](#)  
*[in] Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.*
- const uint8\_t \* [client\\_chal](#)  
*[in] ClientChal data, 32 bytes. Can be NULL if mode[0] is 1.*
- uint8\_t \* [client\\_resp](#)  
*[out] Calculated ClientResp will be returned here.*
- const uint8\_t \* [other\\_data](#)  
*[in] OtherData, 13 bytes*
- const uint8\_t \* [otp](#)  
*[in] First 8 bytes of the OTP zone data. Can be NULL is mode[5] is 0.*
- const uint8\_t \* [slot\\_key](#)
- const uint8\_t \* [target\\_key](#)
- struct [atca\\_temp\\_key](#) \* [temp\\_key](#)  
*[in,out] Current state of TempKey. Required if mode[0] or mode[1] are 1.*



### 19.15.1 Detailed Description

Input/output parameters for function [atcah\\_check\\_mac\(\)](#).

### 19.15.2 Field Documentation

#### 19.15.2.1 client\_chal

```
const uint8_t* client_chal
```

[in] ClientChal data, 32 bytes. Can be NULL if mode[0] is 1.

#### 19.15.2.2 client\_resp

```
uint8_t* client_resp
```

[out] Calculated ClientResp will be returned here.

#### 19.15.2.3 key\_id

```
uint16_t key_id
```

[in] CheckMac command KeyID

#### 19.15.2.4 mode

```
uint8_t mode
```

[in] CheckMac command Mode

#### 19.15.2.5 other\_data

```
const uint8_t* other_data
```

[in] OtherData, 13 bytes

### 19.15.2.6 otp

```
const uint8_t* otp
```

[in] First 8 bytes of the OTP zone data. Can be NULL is mode[5] is 0.

### 19.15.2.7 slot\_key

```
const uint8_t* slot_key
```

[in] 32 byte key value in the slot specified by slot\_id. Can be NULL if mode[1] is 1.

### 19.15.2.8 sn

```
const uint8_t* sn
```

[in] Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.

### 19.15.2.9 target\_key

```
const uint8_t* target_key
```

[in] If this is not NULL, it assumes CheckMac copy is enabled for the specified key\_id (ReadKey=0). If key\_id is even, this should be the 32-byte key value for the slot key\_id+1, otherwise this should be set to slot\_key.

### 19.15.2.10 temp\_key

```
struct atca_temp_key* temp_key
```

[in,out] Current state of TempKey. Required if mode[0] or mode[1] are 1.

## 19.16 atca\_command Struct Reference

[atca\\_command](#) is the C object backing ATCACommand.

```
#include <atca_command.h>
```

### Data Fields

- [ATCADeviceType](#) dt
- [uint8\\_t](#) clock\_divider
- [uint16\\_t](#) execution\_time\_msec

### 19.16.1 Detailed Description

[atca\\_command](#) is the C object backing ATCACommand.

### 19.16.2 Field Documentation

#### 19.16.2.1 clock\_divider

`uint8_t clock_divider`

#### 19.16.2.2 dt

`ATCADeviceType dt`

#### 19.16.2.3 execution\_time\_msec

`uint16_t execution_time_msec`

## 19.17 atca\_decrypt\_in\_out Struct Reference

Input/output parameters for function `atca_decrypt()`.

```
#include <atca_host.h>
```

### Data Fields

- `uint8_t * crypto\_data`  
*[in,out] Pointer to 32-byte data. Input encrypted data from Read command (Contents field), output decrypted.*
- `struct atca\_temp\_key * temp\_key`  
*[in,out] Pointer to TempKey structure.*

### 19.17.1 Detailed Description

Input/output parameters for function `atca_decrypt()`.

## 19.18 atca\_derive\_key\_in\_out Struct Reference

Input/output parameters for function [atcah\\_derive\\_key\(\)](#).

```
#include <atca_host.h>
```

### Data Fields

- `uint8_t mode`  
*Mode (param 1) of the derive key command.*
- `uint16_t target_key_id`  
*Key ID (param 2) of the target slot to run the command on.*
- `const uint8_t * sn`  
*Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.*
- `const uint8_t * parent_key`  
*Parent key to be used in the derive key calculation (32 bytes).*
- `uint8_t * target_key`  
*Derived key will be returned here (32 bytes).*
- `struct atca_temp_key * temp_key`  
*Current state of TempKey.*

### 19.18.1 Detailed Description

Input/output parameters for function [atcah\\_derive\\_key\(\)](#).

### 19.18.2 Field Documentation

#### 19.18.2.1 mode

```
uint8_t mode
```

Mode (param 1) of the derive key command.

#### 19.18.2.2 parent\_key

```
const uint8_t* parent_key
```

Parent key to be used in the derive key calculation (32 bytes).

### 19.18.2.3 sn

```
const uint8_t* sn
```

Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.

### 19.18.2.4 target\_key

```
uint8_t* target_key
```

Derived key will be returned here (32 bytes).

### 19.18.2.5 target\_key\_id

```
uint16_t target_key_id
```

Key ID (param 2) of the target slot to run the command on.

### 19.18.2.6 temp\_key

```
struct atca_temp_key* temp_key
```

Current state of TempKey.

## 19.19 atca\_derive\_key\_mac\_in\_out Struct Reference

Input/output parameters for function [atcah\\_derive\\_key\\_mac\(\)](#).

```
#include <atca_host.h>
```

### Data Fields

- `uint8_t mode`  
*Mode (param 1) of the derive key command.*
- `uint16_t target_key_id`  
*Key ID (param 2) of the target slot to run the command on.*
- `const uint8_t * sn`  
*Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.*
- `const uint8_t * parent_key`  
*Parent key to be used in the derive key calculation (32 bytes).*
- `uint8_t * mac`  
*DeriveKey MAC will be returned here.*

### 19.19.1 Detailed Description

Input/output parameters for function [atcah\\_derive\\_key\\_mac\(\)](#).

### 19.19.2 Field Documentation

#### 19.19.2.1 mac

```
uint8_t* mac
```

DeriveKey MAC will be returned here.

#### 19.19.2.2 mode

```
uint8_t mode
```

Mode (param 1) of the derive key command.

#### 19.19.2.3 parent\_key

```
const uint8_t* parent_key
```

Parent key to be used in the derive key calculation (32 bytes).

#### 19.19.2.4 sn

```
const uint8_t* sn
```

Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.

#### 19.19.2.5 target\_key\_id

```
uint16_t target_key_id
```

Key ID (param 2) of the target slot to run the command on.

## 19.20 atca\_device Struct Reference

[atca\\_device](#) is the C object backing ATCADevice. See the [atca\\_device.h](#) file for details on the ATCADevice methods

```
#include <atca_device.h>
```

### Data Fields

- [ATCACommand mCommands](#)  
*Command set for a given CryptoAuth device.*
- [ATCAIface mIface](#)  
*Physical interface.*
- `uint8_t` [session\\_state](#)
- `uint16_t` [session\\_counter](#)
- `uint16_t` [session\\_key\\_id](#)
- `uint8_t *` [session\\_key](#)
- `uint8_t` [session\\_key\\_len](#)

### 19.20.1 Detailed Description

[atca\\_device](#) is the C object backing ATCADevice. See the [atca\\_device.h](#) file for details on the ATCADevice methods

### 19.20.2 Field Documentation

#### 19.20.2.1 mCommands

[ATCACommand](#) mCommands

Command set for a given CryptoAuth device.

#### 19.20.2.2 mIface

[ATCAIface](#) mIface

Physical interface.

#### 19.20.2.3 session\_counter

`uint16_t` session\_counter

Secure Session Message Count

### 19.20.2.4 session\_key

uint8\_t\* session\_key

Session Key

### 19.20.2.5 session\_key\_id

uint16\_t session\_key\_id

Key ID used for a secure session

### 19.20.2.6 session\_key\_len

uint8\_t session\_key\_len

Length of key used for the session in bytes

### 19.20.2.7 session\_state

uint8\_t session\_state

Secure Session State

## 19.21 atca\_gen\_dig\_in\_out Struct Reference

Input/output parameters for function [atca\\_gen\\_dig\(\)](#).

```
#include <atca_host.h>
```

### Data Fields

- uint8\_t [zone](#)  
*[in]* Zone/Param1 for the GenDig command
- uint16\_t [key\\_id](#)  
*[in]* KeyId/Param2 for the GenDig command
- uint16\_t [slot\\_conf](#)  
*[in]* Slot config for the GenDig command
- uint16\_t [key\\_conf](#)  
*[in]* Key config for the GenDig command
- uint8\_t [slot\\_locked](#)  
*[in]* slot locked for the GenDig command
- uint32\_t [counter](#)  
*[in]* counter for the GenDig command
- bool [is\\_key\\_nomac](#)  
*[in]* Set to true if the slot pointed to be key\_id has the SotConfig.NoMac bit set
- const uint8\_t \* [sn](#)  
*[in]* Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.
- const uint8\_t \* [stored\\_value](#)  
*[in]* 32-byte slot value, config block, OTP block as specified by the Zone/KeyId parameters
- const uint8\_t \* [other\\_data](#)  
*[in]* 32-byte value for shared nonce zone, 4-byte value if is\_key\_nomac is true, ignored and/or NULL otherwise
- struct [atca\\_temp\\_key](#) \* [temp\\_key](#)  
*[inout]* Current state of TempKey



### 19.21.1 Detailed Description

Input/output parameters for function [atcah\\_gen\\_dig\(\)](#).

### 19.21.2 Field Documentation

#### 19.21.2.1 counter

```
uint32_t counter
```

[in] counter for the GenDig command

#### 19.21.2.2 is\_key\_nomac

```
bool is_key_nomac
```

[in] Set to true if the slot pointed to be key\_id has the SotConfig.NoMac bit set

#### 19.21.2.3 key\_conf

```
uint16_t key_conf
```

[in] Key config for the GenDig command

#### 19.21.2.4 key\_id

```
uint16_t key_id
```

[in] KeyId/Param2 for the GenDig command

#### 19.21.2.5 other\_data

```
const uint8_t* other_data
```

[in] 32-byte value for shared nonce zone, 4-byte value if is\_key\_nomac is true, ignored and/or NULL otherwise

### 19.21.2.6 slot\_conf

```
uint16_t slot_conf
```

[in] Slot config for the GenDig command

### 19.21.2.7 slot\_locked

```
uint8_t slot_locked
```

[in] slot locked for the GenDig command

### 19.21.2.8 sn

```
const uint8_t* sn
```

[in] Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.

### 19.21.2.9 stored\_value

```
const uint8_t* stored_value
```

[in] 32-byte slot value, config block, OTP block as specified by the Zone/KeyId parameters

### 19.21.2.10 temp\_key

```
struct atca_temp_key* temp_key
```

[inout] Current state of TempKey

### 19.21.2.11 zone

```
uint8_t zone
```

[in] Zone/Param1 for the GenDig command

## 19.22 atca\_gen\_key\_in\_out Struct Reference

Input/output parameters for calculating the PubKey digest put into TempKey by the GenKey command with the [atcah\\_gen\\_key\\_msg\(\)](#) function.

```
#include <atca_host.h>
```

### Data Fields

- `uint8_t mode`  
*[in] GenKey Mode*
- `uint16_t key_id`  
*[in] GenKey KeyID*
- `const uint8_t * public_key`  
*[in] Public key to be used in the PubKey digest. X and Y integers in big-endian format. 64 bytes for P256 curve.*
- `size_t public_key_size`  
*[in] Total number of bytes in the public key. 64 bytes for P256 curve.*
- `const uint8_t * other_data`  
*[in] 3 bytes required when bit 4 of the mode is set. Can be NULL otherwise.*
- `const uint8_t * sn`  
*[in] Device serial number SN[0:8] (9 bytes). Only SN[0:1] and SN[8] are required though.*
- `struct atca_temp_key * temp_key`  
*[in,out] As input the current state of TempKey. As output, the resulting PubKey digest.*

### 19.22.1 Detailed Description

Input/output parameters for calculating the PubKey digest put into TempKey by the GenKey command with the [atcah\\_gen\\_key\\_msg\(\)](#) function.

### 19.22.2 Field Documentation

#### 19.22.2.1 key\_id

```
uint16_t key_id
```

*[in] GenKey KeyID*

#### 19.22.2.2 mode

```
uint8_t mode
```

*[in] GenKey Mode*

## 19.23 atca\_hmac\_in\_out Struct Reference

---

### 19.22.2.3 other\_data

```
const uint8_t* other_data
```

[in] 3 bytes required when bit 4 of the mode is set. Can be NULL otherwise.

### 19.22.2.4 public\_key

```
const uint8_t* public_key
```

[in] Public key to be used in the PubKey digest. X and Y integers in big-endian format. 64 bytes for P256 curve.

### 19.22.2.5 public\_key\_size

```
size_t public_key_size
```

[in] Total number of bytes in the public key. 64 bytes for P256 curve.

### 19.22.2.6 sn

```
const uint8_t* sn
```

[in] Device serial number SN[0:8] (9 bytes). Only SN[0:1] and SN[8] are required though.

### 19.22.2.7 temp\_key

```
struct atca_temp_key* temp_key
```

[in,output] As input the current state of TempKey. As output, the resulting PubKEy digest.

## 19.23 atca\_hmac\_in\_out Struct Reference

Input/output parameters for function `atca_hmac()`.

```
#include <atca_host.h>
```

## Data Fields

- `uint8_t mode`  
*[in] Mode parameter used in HMAC command (Param1).*
- `uint16_t key_id`  
*[in] KeyID parameter used in HMAC command (Param2).*
- `const uint8_t * key`  
*[in] Pointer to 32-byte key used to generate HMAC digest.*
- `const uint8_t * otp`  
*[in] Pointer to 11-byte OTP, optionally included in HMAC digest, depending on mode.*
- `const uint8_t * sn`  
*[in] Pointer to 9-byte SN, optionally included in HMAC digest, depending on mode.*
- `uint8_t * response`  
*[out] Pointer to 32-byte SHA-256 HMAC digest.*
- `struct atca_temp_key * temp_key`  
*[in,out] Pointer to TempKey structure.*

### 19.23.1 Detailed Description

Input/output parameters for function `atca_hmac()`.

## 19.24 atca\_iface Struct Reference

`atca_iface` is the C object backing ATCAIface. See the [atca\\_iface.h](#) file for details on the ATCAIface methods

```
#include <atca_iface.h>
```

## Data Fields

- `ATCAIfaceType mType`
- `ATCAIfaceCfg * mIfaceCFG`
- `ATCA_STATUS(* atinit)(void *hal, ATCAIfaceCfg *)`
- `ATCA_STATUS(* atpostinit)(ATCAIface hal)`
- `ATCA_STATUS(* atsend)(ATCAIface hal, uint8_t word_address, uint8_t *txdata, int txlength)`
- `ATCA_STATUS(* atreceive)(ATCAIface hal, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)`
- `ATCA_STATUS(* atwake)(ATCAIface hal)`
- `ATCA_STATUS(* atidle)(ATCAIface hal)`
- `ATCA_STATUS(* atsleep)(ATCAIface hal)`
- `void * hal_data`

### 19.24.1 Detailed Description

`atca_iface` is the C object backing ATCAIface. See the [atca\\_iface.h](#) file for details on the ATCAIface methods

### 19.24.2 Field Documentation

### 19.24.2.1 atidle

`ATCA_STATUS(* atidle) (ATCAIface hal)`

### 19.24.2.2 atinit

`ATCA_STATUS(* atinit) (void *hal, ATCAIfaceCfg *)`

### 19.24.2.3 atpostinit

`ATCA_STATUS(* atpostinit) (ATCAIface hal)`

### 19.24.2.4 atreceive

`ATCA_STATUS(* atreceive) (ATCAIface hal, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)`

### 19.24.2.5 atsend

`ATCA_STATUS(* atsend) (ATCAIface hal, uint8_t word_address, uint8_t *txdata, int txlength)`

### 19.24.2.6 atsleep

`ATCA_STATUS(* atsleep) (ATCAIface hal)`

### 19.24.2.7 atwake

`ATCA_STATUS(* atwake) (ATCAIface hal)`

### 19.24.2.8 hal\_data

`void* hal_data`

### 19.24.2.9 mIfaceCFG

`ATCAIfaceCfg* mIfaceCFG`

### 19.24.2.10 mType

`ATCAIfaceType mType`

## 19.25 atca\_include\_data\_in\_out Struct Reference

Input / output parameters for function `atca_include_data()`.

```
#include <atca_host.h>
```

### Data Fields

- `uint8_t * p_temp`  
*[out] pointer to output buffer*
- `const uint8_t * otp`  
*[in] pointer to one-time-programming data*
- `const uint8_t * sn`  
*[in] pointer to serial number data*
- `uint8_t mode`

### 19.25.1 Detailed Description

Input / output parameters for function `atca_include_data()`.

### 19.25.2 Field Documentation

#### 19.25.2.1 mode

`uint8_t mode`

## 19.26 atca\_io\_decrypt\_in\_out Struct Reference

```
#include <atca_host.h>
```

### Data Fields

- `const uint8_t * io_key`  
*IO protection key (32 bytes).*
- `const uint8_t * out_nonce`  
*OutNonce returned from command (32 bytes).*
- `uint8_t * data`  
*As input, encrypted data. As output, decrypted data.*
- `size_t data_size`  
*Size of data in bytes (32 or 64).*

### 19.26.1 Field Documentation

#### 19.26.1.1 data

```
uint8_t* data
```

As input, encrypted data. As output, decrypted data.

#### 19.26.1.2 data\_size

```
size_t data_size
```

Size of data in bytes (32 or 64).

#### 19.26.1.3 io\_key

```
const uint8_t* io_key
```

IO protection key (32 bytes).

#### 19.26.1.4 out\_nonce

```
const uint8_t* out_nonce
```

OutNonce returned from command (32 bytes).



## 19.27 atca\_jwt\_t Struct Reference

Structure to hold metadata information about the jwt being built.

```
#include <atca_jwt.h>
```

### Data Fields

- char \* [buf](#)
- uint16\_t [buflen](#)
- uint16\_t [cur](#)

### 19.27.1 Detailed Description

Structure to hold metadata information about the jwt being built.

### 19.27.2 Field Documentation

#### 19.27.2.1 buf

```
char* buf
```

#### 19.27.2.2 buflen

```
uint16_t buflen
```

#### 19.27.2.3 cur

```
uint16_t cur
```

## 19.28 atca\_mac\_in\_out Struct Reference

Input/output parameters for function `atca_mac()`.

```
#include <atca_host.h>
```

### Data Fields

- `uint8_t mode`  
*[in]* Mode parameter used in MAC command (Param1).
- `uint16_t key_id`  
*[in]* KeyID parameter used in MAC command (Param2).
- `const uint8_t * challenge`  
*[in]* Pointer to 32-byte Challenge data used in MAC command, depending on mode.
- `const uint8_t * key`  
*[in]* Pointer to 32-byte key used to generate MAC digest.
- `const uint8_t * otp`  
*[in]* Pointer to 11-byte OTP, optionally included in MAC digest, depending on mode.
- `const uint8_t * sn`  
*[in]* Pointer to 9-byte SN, optionally included in MAC digest, depending on mode.
- `uint8_t * response`  
*[out]* Pointer to 32-byte SHA-256 digest (MAC).
- `struct atca_temp_key * temp_key`  
*[in,out]* Pointer to TempKey structure.

### 19.28.1 Detailed Description

Input/output parameters for function `atca_mac()`.

## 19.29 atca\_nonce\_in\_out Struct Reference

Input/output parameters for function `atca_nonce()`.

```
#include <atca_host.h>
```

### Data Fields

- `uint8_t mode`  
*[in]* Mode parameter used in Nonce command (Param1).
- `uint16_t zero`  
*[in]* Zero parameter used in Nonce command (Param2).
- `const uint8_t * num_in`  
*[in]* Pointer to 20-byte NumIn data used in Nonce command.
- `const uint8_t * rand_out`  
*[in]* Pointer to 32-byte RandOut data from Nonce command.
- `struct atca_temp_key * temp_key`  
*[in,out]* Pointer to TempKey structure.

### 19.29.1 Detailed Description

Input/output parameters for function `atca_nonce()`.

## 19.30 atca\_secureboot\_enc\_in\_out Struct Reference

```
#include <atca_host.h>
```

### Data Fields

- `const uint8_t * io_key`  
*IO protection key value (32 bytes)*
- `const struct atca_temp_key * temp_key`  
*Current value of TempKey.*
- `const uint8_t * digest`  
*Plaintext digest as input.*
- `uint8_t * hashed_key`  
*Calculated key is returned here (32 bytes)*
- `uint8_t * digest_enc`  
*Encrypted (ciphertext) digest is return here (32 bytes)*

### 19.30.1 Field Documentation

#### 19.30.1.1 digest

```
const uint8_t* digest
```

Plaintext digest as input.

#### 19.30.1.2 digest\_enc

```
uint8_t* digest_enc
```

Encrypted (ciphertext) digest is return here (32 bytes)

#### 19.30.1.3 hashed\_key

```
uint8_t* hashed_key
```

Calculated key is returned here (32 bytes)

## 19.31 atca\_secureboot\_mac\_in\_out Struct Reference

---

### 19.30.1.4 io\_key

```
const uint8_t* io_key
```

IO protection key value (32 bytes)

### 19.30.1.5 temp\_key

```
const struct atca_temp_key* temp_key
```

Current value of TempKey.

## 19.31 atca\_secureboot\_mac\_in\_out Struct Reference

```
#include <atca_host.h>
```

### Data Fields

- uint8\_t [mode](#)  
*SecureBoot mode (param1)*
- uint16\_t [param2](#)  
*SecureBoot param2.*
- uint16\_t [secure\\_boot\\_config](#)  
*SecureBootConfig value from configuration zone.*
- const uint8\_t \* [hashed\\_key](#)  
*Hashed key. SHA256(IO Protection Key | TempKey)*
- const uint8\_t \* [digest](#)  
*Digest (unencrypted)*
- const uint8\_t \* [signature](#)  
*Signature (can be NULL if not required)*
- uint8\_t \* [mac](#)  
*MAC is returned here.*

### 19.31.1 Field Documentation

#### 19.31.1.1 digest

```
const uint8_t* digest
```

Digest (unencrypted)

**19.31.1.2 hashed\_key**

```
const uint8_t* hashed_key
```

Hashed key. SHA256(IO Protection Key | TempKey)

**19.31.1.3 mac**

```
uint8_t* mac
```

MAC is returned here.

**19.31.1.4 mode**

```
uint8_t mode
```

SecureBoot mode (param1)

**19.31.1.5 param2**

```
uint16_t param2
```

SecureBoot param2.

**19.31.1.6 secure\_boot\_config**

```
uint16_t secure_boot_config
```

SecureBootConfig value from configuration zone.

**19.31.1.7 signature**

```
const uint8_t* signature
```

Signature (can be NULL if not required)

## 19.32 atca\_sha256\_ctx Struct Reference

```
#include <calib_basic.h>
```

### Data Fields

- uint32\_t [total\\_msg\\_size](#)  
*Total number of message bytes processed.*
- uint32\_t [block\\_size](#)  
*Number of bytes in current block.*
- uint8\_t [block](#) [[ATCA\\_SHA256\\_BLOCK\\_SIZE](#) \*2]  
*Unprocessed message storage.*

### 19.32.1 Field Documentation

#### 19.32.1.1 block

```
uint8_t block[ATCA_SHA256_BLOCK_SIZE *2]
```

Unprocessed message storage.

#### 19.32.1.2 block\_size

```
uint32_t block_size
```

Number of bytes in current block.

#### 19.32.1.3 total\_msg\_size

```
uint32_t total_msg_size
```

Total number of message bytes processed.

## 19.33 atca\_sign\_internal\_in\_out Struct Reference

Input/output parameters for calculating the message and digest used by the Sign(internal) command. Used with the [atcah\\_sign\\_internal\\_msg\(\)](#) function.

```
#include <atca_host.h>
```

## Data Fields

- `uint8_t mode`  
*[in] Sign Mode*
- `uint16_t key_id`  
*[in] Sign KeyID*
- `uint16_t slot_config`  
*[in] SlotConfig[TempKeyFlags.keyId]*
- `uint16_t key_config`  
*[in] KeyConfig[TempKeyFlags.keyId]*
- `uint8_t use_flag`  
*[in] UseFlag[TempKeyFlags.keyId], 0x00 for slots 8 and above and for ATECC508A*
- `uint8_t update_count`  
*[in] UpdateCount[TempKeyFlags.keyId], 0x00 for slots 8 and above and for ATECC508A*
- `bool is_slot_locked`  
*[in] Is TempKeyFlags.keyId slot locked.*
- `bool for_invalidate`  
*[in] Set to true if this will be used for the Verify(Invalidate) command.*
- `const uint8_t * sn`  
*[in] Device serial number SN[0:8] (9 bytes)*
- `const struct atca_temp_key * temp_key`  
*[in] The current state of TempKey.*
- `uint8_t * message`  
*[out] Full 55 byte message the Sign(internal) command will build. Can be NULL if not required.*
- `uint8_t * verify_other_data`  
*[out] The 19 byte OtherData bytes to be used with the Verify(In/Validate) command. Can be NULL if not required.*
- `uint8_t * digest`  
*[out] SHA256 digest of the full 55 byte message. Can be NULL if not required.*

### 19.33.1 Detailed Description

Input/output parameters for calculating the message and digest used by the Sign(internal) command. Used with the `atcah_sign_internal_msg()` function.

### 19.33.2 Field Documentation

#### 19.33.2.1 digest

```
uint8_t* digest
```

*[out]* SHA256 digest of the full 55 byte message. Can be NULL if not required.

### 19.33.2.2 for\_invalidate

bool for\_invalidate

[in] Set to true if this will be used for the Verify(Invalidate) command.

### 19.33.2.3 is\_slot\_locked

bool is\_slot\_locked

[in] Is TempKeyFlags.keyId slot locked.

### 19.33.2.4 key\_config

uint16\_t key\_config

[in] KeyConfig[TempKeyFlags.keyId]

### 19.33.2.5 key\_id

uint16\_t key\_id

[in] Sign KeyID

### 19.33.2.6 message

uint8\_t\* message

[out] Full 55 byte message the Sign(internal) command will build. Can be NULL if not required.

### 19.33.2.7 mode

uint8\_t mode

[in] Sign Mode



**19.33.2.8 slot\_config**

```
uint16_t slot_config
```

[in] SlotConfig[TempKeyFlags.keyId]

**19.33.2.9 sn**

```
const uint8_t* sn
```

[in] Device serial number SN[0:8] (9 bytes)

**19.33.2.10 temp\_key**

```
const struct atca_temp_key* temp_key
```

[in] The current state of TempKey.

**19.33.2.11 update\_count**

```
uint8_t update_count
```

[in] UpdateCount[TempKeyFlags.keyId], 0x00 for slots 8 and above and for ATECC508A

**19.33.2.12 use\_flag**

```
uint8_t use_flag
```

[in] UseFlag[TempKeyFlags.keyId], 0x00 for slots 8 and above and for ATECC508A

**19.33.2.13 verify\_other\_data**

```
uint8_t* verify_other_data
```

[out] The 19 byte OtherData bytes to be used with the Verify(In/Validate) command. Can be NULL if not required.

## 19.34 atca\_temp\_key Struct Reference

Structure to hold TempKey fields.

```
#include <atca_host.h>
```

### Data Fields

- `uint8_t value [ATCA_KEY_SIZE *2]`  
*Value of TempKey (64 bytes for ATECC608 only)*
- `unsigned key_id: 4`  
*If TempKey was derived from a slot or transport key (GenDig or GenKey), that key ID is saved here.*
- `unsigned source_flag: 1`  
*Indicates id TempKey started from a random nonce (0) or not (1).*
- `unsigned gen_dig_data: 1`  
*TempKey was derived from the GenDig command.*
- `unsigned gen_key_data: 1`  
*TempKey was derived from the GenKey command (ATECC devices only).*
- `unsigned no_mac_flag: 1`  
*TempKey was derived from a key that has the NoMac bit set preventing the use of the MAC command. Known as CheckFlag in ATSHA devices).*
- `unsigned valid: 1`  
*TempKey is valid.*
- `uint8_t is_64`  
*TempKey has 64 bytes of valid data.*

### 19.34.1 Detailed Description

Structure to hold TempKey fields.

### 19.34.2 Field Documentation

#### 19.34.2.1 gen\_dig\_data

```
unsigned gen_dig_data
```

TempKey was derived from the GenDig command.

#### 19.34.2.2 gen\_key\_data

```
unsigned gen_key_data
```

TempKey was derived from the GenKey command (ATECC devices only).

**19.34.2.3 is\_64**

```
uint8_t is_64
```

TempKey has 64 bytes of valid data.

**19.34.2.4 key\_id**

```
unsigned key_id
```

If TempKey was derived from a slot or transport key (GenDig or GenKey), that key ID is saved here.

**19.34.2.5 no\_mac\_flag**

```
unsigned no_mac_flag
```

TempKey was derived from a key that has the NoMac bit set preventing the use of the MAC command. Known as CheckFlag in ATSHA devices).

**19.34.2.6 source\_flag**

```
unsigned source_flag
```

Indicates id TempKey started from a random nonce (0) or not (1).

**19.34.2.7 valid**

```
unsigned valid
```

TempKey is valid.

**19.34.2.8 value**

```
uint8_t value[ATCA_KEY_SIZE *2]
```

Value of TempKey (64 bytes for ATECC608 only)

## 19.35 atca\_verify\_in\_out Struct Reference

Input/output parameters for function atcah\_verify().

```
#include <atca_host.h>
```

### Data Fields

- uint16\_t [curve\\_type](#)  
*[in]* Curve type used in Verify command (Param2).
- const uint8\_t \* [signature](#)  
*[in]* Pointer to ECDSA signature to be verified
- const uint8\_t \* [public\\_key](#)  
*[in]* Pointer to the public key to be used for verification
- struct [atca\\_temp\\_key](#) \* [temp\\_key](#)  
*[in,out]* Pointer to TempKey structure.

### 19.35.1 Detailed Description

Input/output parameters for function atcah\_verify().

## 19.36 atca\_verify\_mac Struct Reference

```
#include <atca_host.h>
```

### Data Fields

- uint8\_t [mode](#)  
*Mode (Param1) parameter used in Verify command.*
- uint16\_t [key\\_id](#)  
*KeyID (Param2) used in Verify command.*
- const uint8\_t \* [signature](#)  
*Signature used in Verify command (64 bytes).*
- const uint8\_t \* [other\\_data](#)  
*OtherData used in Verify command (19 bytes).*
- const uint8\_t \* [msg\\_dig\\_buf](#)  
*Message digest buffer (64 bytes).*
- const uint8\_t \* [io\\_key](#)  
*IO protection key value (32 bytes).*
- const uint8\_t \* [sn](#)  
*Serial number (9 bytes).*
- const [atca\\_temp\\_key\\_t](#) \* [temp\\_key](#)  
*TempKey.*
- uint8\_t \* [mac](#)  
*Calculated verification MAC is returned here (32 bytes).*

## 19.36.1 Field Documentation

### 19.36.1.1 io\_key

```
const uint8_t* io_key
```

IO protection key value (32 bytes).

### 19.36.1.2 key\_id

```
uint16_t key_id
```

KeyID (Param2) used in Verify command.

### 19.36.1.3 mac

```
uint8_t* mac
```

Calculated verification MAC is returned here (32 bytes).

### 19.36.1.4 mode

```
uint8_t mode
```

Mode (Param1) parameter used in Verify command.

### 19.36.1.5 msg\_dig\_buf

```
const uint8_t* msg_dig_buf
```

Message digest buffer (64 bytes).

## 19.37 atca\_write\_mac\_in\_out Struct Reference

---

### 19.36.1.6 other\_data

```
const uint8_t* other_data
```

OtherData used in Verify command (19 bytes).

### 19.36.1.7 signature

```
const uint8_t* signature
```

Signature used in Verify command (64 bytes).

### 19.36.1.8 sn

```
const uint8_t* sn
```

Serial number (9 bytes).

### 19.36.1.9 temp\_key

```
const atca_temp_key_t* temp_key
```

TempKey.

## 19.37 atca\_write\_mac\_in\_out Struct Reference

Input/output parameters for function [atcah\\_write\\_auth\\_mac\(\)](#) and [atcah\\_privwrite\\_auth\\_mac\(\)](#).

```
#include <atca_host.h>
```

### Data Fields

- `uint8_t zone`  
*Zone/Param1 for the Write or PrivWrite command.*
- `uint16_t key_id`  
*KeyID/Param2 for the Write or PrivWrite command.*
- `const uint8_t * sn`  
*Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.*
- `const uint8_t * input_data`  
*Data to be encrypted. 32 bytes for Write command, 36 bytes for PrivWrite command.*
- `uint8_t * encrypted_data`  
*Encrypted version of input\_data will be returned here. 32 bytes for Write command, 36 bytes for PrivWrite command.*
- `uint8_t * auth_mac`  
*Write MAC will be returned here. 32 bytes.*
- `struct atca_temp_key * temp_key`  
*Current state of TempKey.*

### 19.37.1 Detailed Description

Input/output parameters for function [atcah\\_write\\_auth\\_mac\(\)](#) and [atcah\\_privwrite\\_auth\\_mac\(\)](#).

### 19.37.2 Field Documentation

#### 19.37.2.1 auth\_mac

```
uint8_t* auth_mac
```

Write MAC will be returned here. 32 bytes.

#### 19.37.2.2 encrypted\_data

```
uint8_t* encrypted_data
```

Encrypted version of input\_data will be returned here. 32 bytes for Write command, 36 bytes for PrivWrite command.

#### 19.37.2.3 input\_data

```
const uint8_t* input_data
```

Data to be encrypted. 32 bytes for Write command, 36 bytes for PrivWrite command.

#### 19.37.2.4 key\_id

```
uint16_t key_id
```

KeyID/Param2 for the Write or PrivWrite command.

#### 19.37.2.5 sn

```
const uint8_t* sn
```

Device serial number SN[0:8]. Only SN[0:1] and SN[8] are required though.

## 19.38 atcacert\_build\_state\_s Struct Reference

---

### 19.37.2.6 temp\_key

```
struct atca_temp_key* temp_key
```

Current state of TempKey.

### 19.37.2.7 zone

```
uint8_t zone
```

Zone/Param1 for the Write or PrivWrite command.

## 19.38 atcacert\_build\_state\_s Struct Reference

```
#include <atcacert_def.h>
```

### Data Fields

- const [atcacert\\_def\\_t](#) \* [cert\\_def](#)  
*Certificate definition for the certificate being rebuilt.*
- [uint8\\_t](#) \* [cert](#)  
*Buffer to contain the rebuilt certificate.*
- [size\\_t](#) \* [cert\\_size](#)  
*Current size of the certificate in bytes.*
- [size\\_t](#) [max\\_cert\\_size](#)  
*Max size of the cert buffer in bytes.*
- [uint8\\_t](#) [is\\_device\\_sn](#)  
*Indicates the structure contains the device SN.*
- [uint8\\_t](#) [device\\_sn](#) [9]  
*Storage for the device SN, when it's found.*

### 19.38.1 Detailed Description

Tracks the state of a certificate as it's being rebuilt from device information.

### 19.38.2 Field Documentation



**19.38.2.1 cert**

```
uint8_t* cert
```

Buffer to contain the rebuilt certificate.

**19.38.2.2 cert\_def**

```
const atcacert_def_t* cert_def
```

Certificate definition for the certificate being rebuilt.

**19.38.2.3 cert\_size**

```
size_t* cert_size
```

Current size of the certificate in bytes.

**19.38.2.4 device\_sn**

```
uint8_t device_sn[9]
```

Storage for the device SN, when it's found.

**19.38.2.5 is\_device\_sn**

```
uint8_t is_device_sn
```

Indicates the structure contains the device SN.

**19.38.2.6 max\_cert\_size**

```
size_t max_cert_size
```

Max size of the cert buffer in bytes.

## 19.39 atcacert\_cert\_element\_s Struct Reference

```
#include <atcacert_def.h>
```

### Data Fields

- char [id](#) [25]  
*ID identifying this element.*
- [atcacert\\_device\\_loc\\_t](#) [device\\_loc](#)  
*Location in the device for the element.*
- [atcacert\\_cert\\_loc\\_t](#) [cert\\_loc](#)  
*Location in the certificate template for the element.*
- [atcacert\\_transform\\_t](#) [transforms](#) [2]  
*List of transforms from device to cert for this element.*

### 19.39.1 Detailed Description

Defines a generic dynamic element for a certificate including the device and template locations.

### 19.39.2 Field Documentation

#### 19.39.2.1 cert\_loc

```
atcacert\_cert\_loc\_t cert_loc
```

Location in the certificate template for the element.

#### 19.39.2.2 device\_loc

```
atcacert\_device\_loc\_t device_loc
```

Location in the device for the element.

#### 19.39.2.3 id

```
char id[25]
```

ID identifying this element.

#### 19.39.2.4 transforms

```
atcacert_transform_t transforms[2]
```

List of transforms from device to cert for this element.

### 19.40 atcacert\_cert\_loc\_s Struct Reference

```
#include <atcacert_def.h>
```

#### Data Fields

- uint16\_t [offset](#)  
*Byte offset in the certificate template.*
- uint16\_t [count](#)  
*Byte count. Set to 0 if it doesn't exist.*

#### 19.40.1 Detailed Description

Defines a chunk of data in a certificate template.

#### 19.40.2 Field Documentation

##### 19.40.2.1 count

```
uint16_t count
```

Byte count. Set to 0 if it doesn't exist.

##### 19.40.2.2 offset

```
uint16_t offset
```

Byte offset in the certificate template.

### 19.41 atcacert\_def\_s Struct Reference

```
#include <atcacert_def.h>
```

## Data Fields

- [atcacert\\_cert\\_type\\_t type](#)  
*Certificate type.*
- [uint8\\_t template\\_id](#)  
*ID for the this certificate definition (4-bit value).*
- [uint8\\_t chain\\_id](#)  
*ID for the certificate chain this definition is a part of (4-bit value).*
- [uint8\\_t private\\_key\\_slot](#)  
*If this is a device certificate template, this is the device slot for the device private key.*
- [atcacert\\_cert\\_sn\\_src\\_t sn\\_source](#)  
*Where the certificate serial number comes from (4-bit value).*
- [atcacert\\_device\\_loc\\_t cert\\_sn\\_dev\\_loc](#)  
*Only applies when sn\_source is SNSRC\_STORED or SNSRC\_STORED\_DYNAMIC. Describes where to get the certificate serial number on the device.*
- [atcacert\\_date\\_format\\_t issue\\_date\\_format](#)  
*Format of the issue date in the certificate.*
- [atcacert\\_date\\_format\\_t expire\\_date\\_format](#)  
*format of the expire date in the certificate.*
- [atcacert\\_cert\\_loc\\_t tbs\\_cert\\_loc](#)  
*Location in the certificate for the TBS (to be signed) portion.*
- [uint8\\_t expire\\_years](#)  
*Number of years the certificate is valid for (5-bit value). 0 means no expiration.*
- [atcacert\\_device\\_loc\\_t public\\_key\\_dev\\_loc](#)  
*Where on the device the public key can be found.*
- [atcacert\\_device\\_loc\\_t comp\\_cert\\_dev\\_loc](#)  
*Where on the device the compressed cert can be found.*
- [atcacert\\_cert\\_loc\\_t std\\_cert\\_elements](#) [STDCERT\_NUM\_ELEMENTS]  
*Where in the certificate template the standard cert elements are inserted.*
- [const atcacert\\_cert\\_element\\_t \\* cert\\_elements](#)  
*Additional certificate elements outside of the standard certificate contents.*
- [uint8\\_t cert\\_elements\\_count](#)  
*Number of additional certificate elements in cert\_elements.*
- [const uint8\\_t \\* cert\\_template](#)  
*Pointer to the actual certificate template data.*
- [uint16\\_t cert\\_template\\_size](#)  
*Size of the certificate template in cert\_template in bytes.*
- [const struct atcacert\\_def\\_s \\* ca\\_cert\\_def](#)  
*Certificate definition of the CA certificate.*

### 19.41.1 Detailed Description

Defines a certificate and all the pieces to work with it.

If any of the standard certificate elements (std\_cert\_elements) are not a part of the certificate definition, set their count to 0 to indicate their absence.

### 19.41.2 Field Documentation

#### 19.41.2.1 ca\_cert\_def

```
const struct atcacert_def_s* ca_cert_def
```

Certificate definition of the CA certificate.

#### 19.41.2.2 cert\_elements

```
const atcacert_cert_element_t* cert_elements
```

Additional certificate elements outside of the standard certificate contents.

#### 19.41.2.3 cert\_elements\_count

```
uint8_t cert_elements_count
```

Number of additional certificate elements in cert\_elements.

#### 19.41.2.4 cert\_sn\_dev\_loc

```
atcacert_device_loc_t cert_sn_dev_loc
```

Only applies when sn\_source is SNSRC\_STORED or SNSRC\_STORED\_DYNAMIC. Describes where to get the certificate serial number on the device.

#### 19.41.2.5 cert\_template

```
const uint8_t* cert_template
```

Pointer to the actual certificate template data.

#### 19.41.2.6 cert\_template\_size

```
uint16_t cert_template_size
```

Size of the certificate template in cert\_template in bytes.

### 19.41.2.7 chain\_id

`uint8_t chain_id`

ID for the certificate chain this definition is a part of (4-bit value).

### 19.41.2.8 comp\_cert\_dev\_loc

`atcacert_device_loc_t comp_cert_dev_loc`

Where on the device the compressed cert can be found.

### 19.41.2.9 expire\_date\_format

`atcacert_date_format_t expire_date_format`

format of the expire date in the certificate.

### 19.41.2.10 expire\_years

`uint8_t expire_years`

Number of years the certificate is valid for (5-bit value). 0 means no expiration.

### 19.41.2.11 issue\_date\_format

`atcacert_date_format_t issue_date_format`

Format of the issue date in the certificate.

### 19.41.2.12 private\_key\_slot

`uint8_t private_key_slot`

If this is a device certificate template, this is the device slot for the device private key.

**19.41.2.13 public\_key\_dev\_loc**

`atcacert_device_loc_t` `public_key_dev_loc`

Where on the device the public key can be found.

**19.41.2.14 sn\_source**

`atcacert_cert_sn_src_t` `sn_source`

Where the certificate serial number comes from (4-bit value).

**19.41.2.15 std\_cert\_elements**

`atcacert_cert_loc_t` `std_cert_elements[STDCERT_NUM_ELEMENTS]`

Where in the certificate template the standard cert elements are inserted.

**19.41.2.16 tbs\_cert\_loc**

`atcacert_cert_loc_t` `tbs_cert_loc`

Location in the certificate for the TBS (to be signed) portion.

**19.41.2.17 template\_id**

`uint8_t` `template_id`

ID for the this certificate definition (4-bit value).

**19.41.2.18 type**

`atcacert_cert_type_t` `type`

Certificate type.

## 19.42 atcacert\_device\_loc\_s Struct Reference

```
#include <atcacert_def.h>
```

### Data Fields

- [atcacert\\_device\\_zone\\_t zone](#)  
*Zone in the device.*
- [uint8\\_t slot](#)  
*Slot within the data zone. Only applies if zone is DEVZONE\_DATA.*
- [uint8\\_t is\\_genkey](#)  
*If true, use GenKey command to get the contents instead of Read.*
- [uint16\\_t offset](#)  
*Byte offset in the zone.*
- [uint16\\_t count](#)  
*Byte count.*

### 19.42.1 Detailed Description

Defines a chunk of data in an ATECC device.

### 19.42.2 Field Documentation

#### 19.42.2.1 count

```
uint16_t count
```

Byte count.

#### 19.42.2.2 is\_genkey

```
uint8_t is_genkey
```

If true, use GenKey command to get the contents instead of Read.

#### 19.42.2.3 offset

```
uint16_t offset
```

Byte offset in the zone.



#### 19.42.2.4 slot

```
uint8_t slot
```

Slot within the data zone. Only applies if zone is DEVZONE\_DATA.

#### 19.42.2.5 zone

```
atcacert_device_zone_t zone
```

Zone in the device.

### 19.43 atcacert\_tm\_utc\_s Struct Reference

```
#include <atcacert_date.h>
```

#### Data Fields

- int [tm\\_sec](#)
- int [tm\\_min](#)
- int [tm\\_hour](#)
- int [tm\\_mday](#)
- int [tm\\_mon](#)
- int [tm\\_year](#)

#### 19.43.1 Detailed Description

Holds a broken-down date in UTC. Mimics `atcacert_tm_utc_t` from `time.h`.

#### 19.43.2 Field Documentation

##### 19.43.2.1 tm\_hour

```
int tm_hour
```

##### 19.43.2.2 tm\_mday

```
int tm_mday
```

### 19.43.2.3 tm\_min

```
int tm_min
```

### 19.43.2.4 tm\_mon

```
int tm_mon
```

### 19.43.2.5 tm\_sec

```
int tm_sec
```

### 19.43.2.6 tm\_year

```
int tm_year
```

## 19.44 ATCAHAL\_t Struct Reference

an intermediary data structure to allow the HAL layer to point the standard API functions used by the upper layers to the HAL implementation for the interface. This isolates the upper layers and loosely couples the ATCAIface object from the physical implementation.

```
#include <atca_hal.h>
```

### Data Fields

- [ATCA\\_STATUS\(\\* halinit\)](#)(void \*hal, [ATCAIfaceCfg](#) \*cfg)
- [ATCA\\_STATUS\(\\* halpostinit\)](#)([ATCAIface](#) iface)
- [ATCA\\_STATUS\(\\* halsend\)](#)([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)
- [ATCA\\_STATUS\(\\* halreceive\)](#)([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)
- [ATCA\\_STATUS\(\\* halwake\)](#)([ATCAIface](#) iface)
- [ATCA\\_STATUS\(\\* halidle\)](#)([ATCAIface](#) iface)
- [ATCA\\_STATUS\(\\* halsleep\)](#)([ATCAIface](#) iface)
- [ATCA\\_STATUS\(\\* halrelease\)](#)(void \*hal\_data)
- void \* [hal\\_data](#)

### 19.44.1 Detailed Description

an intermediary data structure to allow the HAL layer to point the standard API functions used by the upper layers to the HAL implementation for the interface. This isolates the upper layers and loosely couples the ATCAIface object from the physical implementation.

## 19.44.2 Field Documentation

### 19.44.2.1 hal\_data

`void* hal_data`

### 19.44.2.2 halidle

`ATCA_STATUS(* halidle) (ATCAIface iface)`

### 19.44.2.3 halinit

`ATCA_STATUS(* halinit) (void *hal, ATCAIfaceCfg *cfg)`

### 19.44.2.4 halpostinit

`ATCA_STATUS(* halpostinit) (ATCAIface iface)`

### 19.44.2.5 halreceive

`ATCA_STATUS(* halreceive) (ATCAIface iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)`

### 19.44.2.6 halrelease

`ATCA_STATUS(* halrelease) (void *hal_data)`

### 19.44.2.7 halsend

`ATCA_STATUS(* halsend) (ATCAIface iface, uint8_t word_address, uint8_t *txdata, int txlength)`

### 19.44.2.8 halsleep

`ATCA_STATUS(* halsleep) (ATCAIface iface)`

### 19.44.2.9 halwake

`ATCA_STATUS(* halwake) (ATCAIface iface)`

## 19.45 atcahid Struct Reference

```
#include <hal_all_platforms_kit_hidapi.h>
```

### Data Fields

- `hid_device * kits` [10]
- `int8_t num_kits_found`
- `hid_device_t kits` [10]

### 19.45.1 Field Documentation

#### 19.45.1.1 kits [1/2]

`hid_device_t kits`

#### 19.45.1.2 kits [2/2]

`hid_device_t kits[10]`

#### 19.45.1.3 num\_kits\_found

`int8_t num_kits_found`

## 19.46 atcal2Cmaster Struct Reference

this is the hal\_data for ATCA HAL for ASF SERCOM

```
#include <hal_linux_i2c_userspace.h>
```

### Data Fields

- int [id](#)
- int [ref\\_ct](#)
- int [bus\\_index](#)
- char [i2c\\_file](#) [16]
- uint8\_t [twi\\_id](#)
- avr32\_twi\_t \* [twi\\_master\\_instance](#)

### 19.46.1 Detailed Description

this is the hal\_data for ATCA HAL for ASF SERCOM

### 19.46.2 Field Documentation

#### 19.46.2.1 bus\_index

```
int bus_index
```

#### 19.46.2.2 i2c\_file

```
char i2c_file[16]
```

#### 19.46.2.3 id

```
int id
```

#### 19.46.2.4 ref\_ct

```
int ref_ct
```

### 19.46.2.5 twi\_id

```
uint8_t twi_id
```

### 19.46.2.6 twi\_master\_instance

```
avr32_twi_t* twi_master_instance
```

## 19.47 ATCAIfaceCfg Struct Reference

```
#include <atca_iface.h>
```

### Data Fields

- [ATCAIfaceType iface\\_type](#)
- [ATCADeviceType devtype](#)
- union {
  - struct {
    - uint8\_t [slave\\_address](#)
    - uint8\_t [bus](#)
    - uint32\_t [baud](#)
  - } [atcai2c](#)
  - struct {
    - uint8\_t [bus](#)
  - } [atcaswi](#)
  - struct {
    - uint8\_t [bus](#)
    - uint8\_t [select\\_pin](#)
    - uint32\_t [baud](#)
  - } [atcaspi](#)
  - struct {
    - int [port](#)
    - uint32\_t [baud](#)
    - uint8\_t [wordsize](#)
    - uint8\_t [parity](#)
    - uint8\_t [stopbits](#)
  - } [atcauart](#)
  - struct {
    - int [idx](#)
    - [ATCAKitType dev\\_interface](#)
    - uint8\_t [dev\\_identity](#)
    - uint32\_t [vid](#)
    - uint32\_t [pid](#)
    - uint32\_t [packetsize](#)
  - } [atcahid](#)
  - struct {
    - [ATCA\\_STATUS\(\\* halinit\)](#)(void \*hal, void \*cfg)
    - [ATCA\\_STATUS\(\\* halpostinit\)](#)(void \*iface)
    - [ATCA\\_STATUS\(\\* halsend\)](#)(void \*iface, uint8\_t word\_address, uint8\_t \*txdata,

```

    int txlength)
    ATCA_STATUS(* halreceive )(void *iface, uint8_t
        word_address, uint8_t *rxdata,
        uint16_t *rxlength)
    ATCA_STATUS(* halwake )(void *iface)
    ATCA_STATUS(* halidle )(void *iface)
    ATCA_STATUS(* halsleep )(void *iface)
        ATCA_STATUS(* halrelease )(void *hal_data)
    } atcacustom
};

```

- uint16\_t wake\_delay
- int rx\_retries
- void \* cfg\_data

## 19.47.1 Field Documentation

### 19.47.1.1 "@1

```
union { ... }
```

### 19.47.1.2 atcacustom

```
struct { ... } atcacustom
```

### 19.47.1.3 atcahid

```
struct { ... } atcahid
```

### 19.47.1.4 atcai2c

```
struct { ... } atcai2c
```

### 19.47.1.5 atcaspi

```
struct { ... } atcaspi
```

### 19.47.1.6 atcaswi

```
struct { ... } atcaswi
```

### 19.47.1.7 atcauart

```
struct { ... } atcauart
```

### 19.47.1.8 baud

```
uint32_t baud
```

### 19.47.1.9 bus

```
uint8_t bus
```

### 19.47.1.10 cfg\_data

```
void* cfg_data
```

### 19.47.1.11 dev\_identity

```
uint8_t dev_identity
```

### 19.47.1.12 dev\_interface

```
ATCAKitType dev_interface
```

### 19.47.1.13 devtype

```
ATCADeviceType devtype
```



**19.47.1.14 halidle**

`ATCA_STATUS(* halidle) (void *iface)`

**19.47.1.15 halinit**

`ATCA_STATUS(* halinit) (void *hal, void *cfg)`

**19.47.1.16 halpostinit**

`ATCA_STATUS(* halpostinit) (void *iface)`

**19.47.1.17 halreceive**

`ATCA_STATUS(* halreceive) (void *iface, uint8_t word_address, uint8_t *rxdata, uint16_t *rxlength)`

**19.47.1.18 halrelease**

`ATCA_STATUS(* halrelease) (void *hal_data)`

**19.47.1.19 halsend**

`ATCA_STATUS(* halsend) (void *iface, uint8_t word_address, uint8_t *txdata, int txlength)`

**19.47.1.20 halsleep**

`ATCA_STATUS(* halsleep) (void *iface)`

**19.47.1.21 halwake**

`ATCA_STATUS(* halwake) (void *iface)`

### 19.47.1.22 idx

int idx

### 19.47.1.23 iface\_type

ATCAIfaceType iface\_type

### 19.47.1.24 packetsize

uint32\_t packetsize

### 19.47.1.25 parity

uint8\_t parity

### 19.47.1.26 pid

uint32\_t pid

### 19.47.1.27 port

int port

### 19.47.1.28 rx\_retries

int rx\_retries

### 19.47.1.29 select\_pin

uint8\_t select\_pin

**19.47.1.30 slave\_address**

```
uint8_t slave_address
```

**19.47.1.31 stopbits**

```
uint8_t stopbits
```

**19.47.1.32 vid**

```
uint32_t vid
```

**19.47.1.33 wake\_delay**

```
uint16_t wake_delay
```

**19.47.1.34 wordsize**

```
uint8_t wordsize
```

## 19.48 ATCAPacket Struct Reference

```
#include <calib_command.h>
```

### Data Fields

- [uint8\\_t \\_reserved](#)
- [uint8\\_t txsize](#)
- [uint8\\_t opcode](#)
- [uint8\\_t param1](#)
- [uint16\\_t param2](#)
- [uint8\\_t data](#) [192]
- [uint8\\_t execTime](#)

### 19.48.1 Field Documentation

### 19.48.1.1 `_reserved`

```
uint8_t _reserved
```

### 19.48.1.2 `data`

```
uint8_t data[192]
```

### 19.48.1.3 `execTime`

```
uint8_t execTime
```

### 19.48.1.4 `opcode`

```
uint8_t opcode
```

### 19.48.1.5 `param1`

```
uint8_t param1
```

### 19.48.1.6 `param2`

```
uint16_t param2
```

### 19.48.1.7 `txsize`

```
uint8_t txsize
```

## 19.49 `atcaSPImaster` Struct Reference

```
#include <hal_linux_spi_userspace.h>
```

## Data Fields

- char [spi\\_file](#) [16]
- int [ref\\_ct](#)

### 19.49.1 Field Documentation

#### 19.49.1.1 [ref\\_ct](#)

```
int ref_ct
```

#### 19.49.1.2 [spi\\_file](#)

```
char spi_file[16]
```

## 19.50 atcaSWImaster Struct Reference

this is the hal\_data for ATCA HAL for ASF SERCOM

```
#include <swi_uart_samd21_asf.h>
```

## Data Fields

- struct usart\_module [usart\\_instance](#)
- int [ref\\_ct](#)
- int [bus\\_index](#)
- struct usart\_sync\_descriptor [USART\\_SWI](#)
- uint32\_t [sercom\\_core\\_freq](#)

### 19.50.1 Detailed Description

this is the hal\_data for ATCA HAL for ASF SERCOM

### 19.50.2 Field Documentation

### 19.50.2.1 bus\_index

```
int bus_index
```

### 19.50.2.2 ref\_ct

```
int ref_ct
```

### 19.50.2.3 sercom\_core\_freq

```
uint32_t sercom_core_freq
```

### 19.50.2.4 usart\_instance

```
struct usart_module usart_instance
```

### 19.50.2.5 USART\_SWI

```
struct usart_sync_descriptor USART_SWI
```

## 19.51 CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_BYTE](#) iv [16]
- [CK\\_BYTE\\_PTR](#) pData
- [CK\\_ULONG](#) length

### 19.51.1 Field Documentation

**19.51.1.1 iv**

`CK_BYTE iv[16]`

**19.51.1.2 length**

`CK_ULONG length`

**19.51.1.3 pData**

`CK_BYTE_PTR pData`

**19.52 CK\_AES\_CCM\_PARAMS Struct Reference**

```
#include <pkcs11t.h>
```

**Data Fields**

- `CK_ULONG ulDataLen`
- `CK_BYTE_PTR pNonce`
- `CK_ULONG ulNonceLen`
- `CK_BYTE_PTR pAAD`
- `CK_ULONG ulAADLen`
- `CK_ULONG ulMACLen`

**19.52.1 Field Documentation****19.52.1.1 pAAD**

`CK_BYTE_PTR pAAD`

**19.52.1.2 pNonce**

`CK_BYTE_PTR pNonce`

## 19.53 CK\_AES\_CTR\_PARAMS Struct Reference

---

### 19.52.1.3 ulAADLen

`CK_ULONG ulAADLen`

### 19.52.1.4 ulDataLen

`CK_ULONG ulDataLen`

### 19.52.1.5 ulMACLen

`CK_ULONG ulMACLen`

### 19.52.1.6 ulNonceLen

`CK_ULONG ulNonceLen`

## 19.53 CK\_AES\_CTR\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_ULONG ulCounterBits`
- `CK_BYTE cb[16]`

### 19.53.1 Field Documentation

#### 19.53.1.1 cb

`CK_BYTE cb[16]`



### 19.53.1.2 ulCounterBits

`CK_ULONG` ulCounterBits

## 19.54 CK\_AES\_GCM\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_BYTE_PTR` pIv
- `CK_ULONG` ulIvLen
- `CK_ULONG` ulIvBits
- `CK_BYTE_PTR` pAAD
- `CK_ULONG` ulAADLen
- `CK_ULONG` ulTagBits

### 19.54.1 Field Documentation

#### 19.54.1.1 pAAD

`CK_BYTE_PTR` pAAD

#### 19.54.1.2 pIv

`CK_BYTE_PTR` pIv

#### 19.54.1.3 ulAADLen

`CK_ULONG` ulAADLen

#### 19.54.1.4 ulIvBits

`CK_ULONG` ulIvBits

### 19.54.1.5 ulIvLen

`CK_ULONG ulIvLen`

### 19.54.1.6 ulTagBits

`CK_ULONG ulTagBits`

## 19.55 CK\_ARIA\_CBC\_ENCRYPT\_DATA\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_BYTE iv` [16]
- `CK_BYTE_PTR pData`
- `CK_ULONG length`

### 19.55.1 Field Documentation

#### 19.55.1.1 iv

`CK_BYTE iv` [16]

#### 19.55.1.2 length

`CK_ULONG length`

#### 19.55.1.3 pData

`CK_BYTE_PTR pData`

## 19.56 CK\_ATTRIBUTE Struct Reference

```
#include <pkcs11t.h>
```

## Data Fields

- [CK\\_ATTRIBUTE\\_TYPE](#) type
- [CK\\_VOID\\_PTR](#) pValue
- [CK\\_ULONG](#) ulValueLen

### 19.56.1 Field Documentation

#### 19.56.1.1 pValue

[CK\\_VOID\\_PTR](#) pValue

#### 19.56.1.2 type

[CK\\_ATTRIBUTE\\_TYPE](#) type

#### 19.56.1.3 ulValueLen

[CK\\_ULONG](#) ulValueLen

## 19.57 CK\_C\_INITIALIZE\_ARGS Struct Reference

```
#include <pkcs11t.h>
```

## Data Fields

- [CK\\_CREATEMUTEX](#) [CreateMutex](#)
- [CK\\_DESTROYMUTEX](#) [DestroyMutex](#)
- [CK\\_LOCKMUTEX](#) [LockMutex](#)
- [CK\\_UNLOCKMUTEX](#) [UnlockMutex](#)
- [CK\\_FLAGS](#) flags
- [CK\\_VOID\\_PTR](#) pReserved

### 19.57.1 Field Documentation

### 19.57.1.1 CreateMutex

CK\_CREATEMUTEX CreateMutex

### 19.57.1.2 DestroyMutex

CK\_DESTROYMUTEX DestroyMutex

### 19.57.1.3 flags

CK\_FLAGS flags

### 19.57.1.4 LockMutex

CK\_LOCKMUTEX LockMutex

### 19.57.1.5 pReserved

CK\_VOID\_PTR pReserved

### 19.57.1.6 UnlockMutex

CK\_UNLOCKMUTEX UnlockMutex

## 19.58 CK\_CAMELLIA\_CBC\_ENCRYPT\_DATA\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- CK\_BYTE iv [16]
- CK\_BYTE\_PTR pData
- CK\_ULONG length

## 19.58.1 Field Documentation

### 19.58.1.1 iv

`CK_BYTE iv[16]`

### 19.58.1.2 length

`CK_ULONG length`

### 19.58.1.3 pData

`CK_BYTE_PTR pData`

## 19.59 CK\_CAMELLIA\_CTR\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_ULONG ulCounterBits`
- `CK_BYTE cb[16]`

## 19.59.1 Field Documentation

### 19.59.1.1 cb

`CK_BYTE cb[16]`

### 19.59.1.2 ulCounterBits

`CK_ULONG ulCounterBits`

## 19.60 CK\_CCM\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_ULONG](#) ulDataLen
- [CK\\_BYTE\\_PTR](#) pNonce
- [CK\\_ULONG](#) ulNonceLen
- [CK\\_BYTE\\_PTR](#) pAAD
- [CK\\_ULONG](#) ulAADLen
- [CK\\_ULONG](#) ulMACLen

### 19.60.1 Field Documentation

#### 19.60.1.1 pAAD

[CK\\_BYTE\\_PTR](#) pAAD

#### 19.60.1.2 pNonce

[CK\\_BYTE\\_PTR](#) pNonce

#### 19.60.1.3 ulAADLen

[CK\\_ULONG](#) ulAADLen

#### 19.60.1.4 ulDataLen

[CK\\_ULONG](#) ulDataLen

#### 19.60.1.5 ulMACLen

[CK\\_ULONG](#) ulMACLen

### 19.60.1.6 ulNonceLen

`CK_ULONG ulNonceLen`

## 19.61 CK\_CMS\_SIG\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_OBJECT_HANDLE certificateHandle`
- `CK_MECHANISM_PTR pSigningMechanism`
- `CK_MECHANISM_PTR pDigestMechanism`
- `CK_UTF8CHAR_PTR pContentType`
- `CK_BYTE_PTR pRequestedAttributes`
- `CK_ULONG ulRequestedAttributesLen`
- `CK_BYTE_PTR pRequiredAttributes`
- `CK_ULONG ulRequiredAttributesLen`

### 19.61.1 Field Documentation

#### 19.61.1.1 certificateHandle

`CK_OBJECT_HANDLE certificateHandle`

#### 19.61.1.2 pContentType

`CK_UTF8CHAR_PTR pContentType`

#### 19.61.1.3 pDigestMechanism

`CK_MECHANISM_PTR pDigestMechanism`

#### 19.61.1.4 pRequestedAttributes

`CK_BYTE_PTR pRequestedAttributes`

## 19.62 CK\_DATE Struct Reference

---

### 19.61.1.5 pRequiredAttributes

[CK\\_BYTE\\_PTR](#) pRequiredAttributes

### 19.61.1.6 pSigningMechanism

[CK\\_MECHANISM\\_PTR](#) pSigningMechanism

### 19.61.1.7 ulRequestedAttributesLen

[CK\\_ULONG](#) ulRequestedAttributesLen

### 19.61.1.8 ulRequiredAttributesLen

[CK\\_ULONG](#) ulRequiredAttributesLen

## 19.62 CK\_DATE Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_CHAR](#) year [4]
- [CK\\_CHAR](#) month [2]
- [CK\\_CHAR](#) day [2]

### 19.62.1 Field Documentation

#### 19.62.1.1 day

[CK\\_CHAR](#) day [2]



### 19.62.1.2 month

`CK_CHAR month[2]`

### 19.62.1.3 year

`CK_CHAR year[4]`

## 19.63 CK\_DES\_CBC\_ENCRYPT\_DATA\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_BYTE iv [8]`
- `CK_BYTE_PTR pData`
- `CK_ULONG length`

### 19.63.1 Field Documentation

#### 19.63.1.1 iv

`CK_BYTE iv[8]`

#### 19.63.1.2 length

`CK_ULONG length`

#### 19.63.1.3 pData

`CK_BYTE_PTR pData`

## 19.64 CK\_DSA\_PARAMETER\_GEN\_PARAM Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_MECHANISM\\_TYPE](#) hash
- [CK\\_BYTE\\_PTR](#) pSeed
- [CK\\_ULONG](#) ulSeedLen
- [CK\\_ULONG](#) ulIndex

### 19.64.1 Field Documentation

#### 19.64.1.1 hash

[CK\\_MECHANISM\\_TYPE](#) hash

#### 19.64.1.2 pSeed

[CK\\_BYTE\\_PTR](#) pSeed

#### 19.64.1.3 ulIndex

[CK\\_ULONG](#) ulIndex

#### 19.64.1.4 ulSeedLen

[CK\\_ULONG](#) ulSeedLen

## 19.65 CK\_ECDH1\_DERIVE\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_EC\\_KDF\\_TYPE](#) kdf
- [CK\\_ULONG](#) ulSharedDataLen
- [CK\\_BYTE\\_PTR](#) pSharedData
- [CK\\_ULONG](#) ulPublicDataLen
- [CK\\_BYTE\\_PTR](#) pPublicData

## 19.65.1 Field Documentation

### 19.65.1.1 kdf

`CK_EC_KDF_TYPE` kdf

### 19.65.1.2 pPublicData

`CK_BYTE_PTR` pPublicData

### 19.65.1.3 pSharedData

`CK_BYTE_PTR` pSharedData

### 19.65.1.4 ulPublicDataLen

`CK_ULONG` ulPublicDataLen

### 19.65.1.5 ulSharedDataLen

`CK_ULONG` ulSharedDataLen

## 19.66 CK\_ECDH2\_DERIVE\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_EC_KDF_TYPE` kdf
- `CK_ULONG` ulSharedDataLen
- `CK_BYTE_PTR` pSharedData
- `CK_ULONG` ulPublicDataLen
- `CK_BYTE_PTR` pPublicData
- `CK_ULONG` ulPrivateDataLen
- `CK_OBJECT_HANDLE` hPrivateData
- `CK_ULONG` ulPublicDataLen2
- `CK_BYTE_PTR` pPublicData2

### 19.66.1 Field Documentation

#### 19.66.1.1 hPrivateKey

[CK\\_OBJECT\\_HANDLE](#) hPrivateKey

#### 19.66.1.2 kdf

[CK\\_EC\\_KDF\\_TYPE](#) kdf

#### 19.66.1.3 pPublicKey

[CK\\_BYTE\\_PTR](#) pPublicKey

#### 19.66.1.4 pPublicKey2

[CK\\_BYTE\\_PTR](#) pPublicKey2

#### 19.66.1.5 pSharedData

[CK\\_BYTE\\_PTR](#) pSharedData

#### 19.66.1.6 ulPrivateKeyLen

[CK\\_ULONG](#) ulPrivateKeyLen

#### 19.66.1.7 ulPublicKeyLen

[CK\\_ULONG](#) ulPublicKeyLen

### 19.66.1.8 ulPublicDataLen2

`CK_ULONG ulPublicDataLen2`

### 19.66.1.9 ulSharedDataLen

`CK_ULONG ulSharedDataLen`

## 19.67 CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_ULONG ulAESKeyBits`
- `CK_EC_KDF_TYPE kdf`
- `CK_ULONG ulSharedDataLen`
- `CK_BYTE_PTR pSharedData`

### 19.67.1 Field Documentation

#### 19.67.1.1 kdf

`CK_EC_KDF_TYPE kdf`

#### 19.67.1.2 pSharedData

`CK_BYTE_PTR pSharedData`

#### 19.67.1.3 ulAESKeyBits

`CK_ULONG ulAESKeyBits`

### 19.67.1.4 ulSharedDataLen

`CK_ULONG` ulSharedDataLen

## 19.68 CK\_ECMQV\_DERIVE\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_EC_KDF_TYPE` kdf
- `CK_ULONG` ulSharedDataLen
- `CK_BYTE_PTR` pSharedData
- `CK_ULONG` ulPublicDataLen
- `CK_BYTE_PTR` pPublicData
- `CK_ULONG` ulPrivateDataLen
- `CK_OBJECT_HANDLE` hPrivateData
- `CK_ULONG` ulPublicDataLen2
- `CK_BYTE_PTR` pPublicData2
- `CK_OBJECT_HANDLE` publicKey

### 19.68.1 Field Documentation

#### 19.68.1.1 hPrivateData

`CK_OBJECT_HANDLE` hPrivateData

#### 19.68.1.2 kdf

`CK_EC_KDF_TYPE` kdf

#### 19.68.1.3 pPublicData

`CK_BYTE_PTR` pPublicData

**19.68.1.4 pPublicData2**

`CK_BYTE_PTR` pPublicData2

**19.68.1.5 pSharedData**

`CK_BYTE_PTR` pSharedData

**19.68.1.6 publicKey**

`CK_OBJECT_HANDLE` publicKey

**19.68.1.7 ulPrivateDataLen**

`CK_ULONG` ulPrivateDataLen

**19.68.1.8 ulPublicDataLen**

`CK_ULONG` ulPublicDataLen

**19.68.1.9 ulPublicDataLen2**

`CK_ULONG` ulPublicDataLen2

**19.68.1.10 ulSharedDataLen**

`CK_ULONG` ulSharedDataLen

**19.69 CK\_FUNCTION\_LIST Struct Reference**

```
#include <pkcs11.h>
```

### Data Fields

- [CK\\_VERSION](#) version

#### 19.69.1 Field Documentation

##### 19.69.1.1 version

[CK\\_VERSION](#) version

## 19.70 CK\_GCM\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_BYTE\\_PTR](#) pIv
- [CK\\_ULONG](#) ulIvLen
- [CK\\_ULONG](#) ulIvBits
- [CK\\_BYTE\\_PTR](#) pAAD
- [CK\\_ULONG](#) ulAADLen
- [CK\\_ULONG](#) ulTagBits

#### 19.70.1 Field Documentation

##### 19.70.1.1 pAAD

[CK\\_BYTE\\_PTR](#) pAAD

##### 19.70.1.2 pIv

[CK\\_BYTE\\_PTR](#) pIv



### 19.70.1.3 ulAADLen

[CK\\_ULONG](#) ulAADLen

### 19.70.1.4 ullvBits

[CK\\_ULONG](#) ullvBits

### 19.70.1.5 ullvLen

[CK\\_ULONG](#) ullvLen

### 19.70.1.6 ulTagBits

[CK\\_ULONG](#) ulTagBits

## 19.71 CK\_GOSTR3410\_DERIVE\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_EC\\_KDF\\_TYPE](#) kdf
- [CK\\_BYTE\\_PTR](#) pPublicData
- [CK\\_ULONG](#) ulPublicDataLen
- [CK\\_BYTE\\_PTR](#) pUKM
- [CK\\_ULONG](#) ulUKMLen

### 19.71.1 Field Documentation

#### 19.71.1.1 kdf

[CK\\_EC\\_KDF\\_TYPE](#) kdf

## 19.72 CK\_GOSTR3410\_KEY\_WRAP\_PARAMS Struct Reference

---

### 19.71.1.2 pPublicData

[CK\\_BYTE\\_PTR](#) pPublicData

### 19.71.1.3 pUKM

[CK\\_BYTE\\_PTR](#) pUKM

### 19.71.1.4 ulPublicDataLen

[CK\\_ULONG](#) ulPublicDataLen

### 19.71.1.5 ulUKMLen

[CK\\_ULONG](#) ulUKMLen

## 19.72 CK\_GOSTR3410\_KEY\_WRAP\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_BYTE\\_PTR](#) pWrapOID
- [CK\\_ULONG](#) ulWrapOIDLen
- [CK\\_BYTE\\_PTR](#) pUKM
- [CK\\_ULONG](#) ulUKMLen
- [CK\\_OBJECT\\_HANDLE](#) hKey

### 19.72.1 Field Documentation

#### 19.72.1.1 hKey

[CK\\_OBJECT\\_HANDLE](#) hKey

### 19.72.1.2 pUKM

[CK\\_BYTE\\_PTR](#) pUKM

### 19.72.1.3 pWrapOID

[CK\\_BYTE\\_PTR](#) pWrapOID

### 19.72.1.4 ulUKMLen

[CK\\_ULONG](#) ulUKMLen

### 19.72.1.5 ulWrapOIDLen

[CK\\_ULONG](#) ulWrapOIDLen

## 19.73 CK\_INFO Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_VERSION](#) cryptokiVersion
- [CK\\_UTF8CHAR](#) manufacturerID [32]
- [CK\\_FLAGS](#) flags
- [CK\\_UTF8CHAR](#) libraryDescription [32]
- [CK\\_VERSION](#) libraryVersion

### 19.73.1 Field Documentation

#### 19.73.1.1 cryptokiVersion

[CK\\_VERSION](#) cryptokiVersion

## 19.74 CK\_KEA\_DERIVE\_PARAMS Struct Reference

---

### 19.73.1.2 flags

`CK_FLAGS` flags

### 19.73.1.3 libraryDescription

`CK_UTF8CHAR` libraryDescription[32]

### 19.73.1.4 libraryVersion

`CK_VERSION` libraryVersion

### 19.73.1.5 manufacturerID

`CK_UTF8CHAR` manufacturerID[32]

## 19.74 CK\_KEA\_DERIVE\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_BBOOL` isSender
- `CK_ULONG` ulRandomLen
- `CK_BYTE_PTR` pRandomA
- `CK_BYTE_PTR` pRandomB
- `CK_ULONG` ulPublicDataLen
- `CK_BYTE_PTR` pPublicData

### 19.74.1 Field Documentation

#### 19.74.1.1 isSender

`CK_BBOOL` isSender

#### 19.74.1.2 pPublicData

[CK\\_BYTE\\_PTR](#) pPublicData

#### 19.74.1.3 pRandomA

[CK\\_BYTE\\_PTR](#) pRandomA

#### 19.74.1.4 pRandomB

[CK\\_BYTE\\_PTR](#) pRandomB

#### 19.74.1.5 ulPublicDataLen

[CK\\_ULONG](#) ulPublicDataLen

#### 19.74.1.6 ulRandomLen

[CK\\_ULONG](#) ulRandomLen

### 19.75 CK\_KEY\_DERIVATION\_STRING\_DATA Struct Reference

```
#include <pkcs11t.h>
```

#### Data Fields

- [CK\\_BYTE\\_PTR](#) pData
- [CK\\_ULONG](#) ulLen

#### 19.75.1 Field Documentation

### 19.75.1.1 pData

[CK\\_BYTE\\_PTR](#) pData

### 19.75.1.2 ulLen

[CK\\_ULONG](#) ulLen

## 19.76 CK\_KEY\_WRAP\_SET\_OAEP\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_BYTE](#) bBC
- [CK\\_BYTE\\_PTR](#) pX
- [CK\\_ULONG](#) ulXLen

### 19.76.1 Field Documentation

#### 19.76.1.1 bBC

[CK\\_BYTE](#) bBC

#### 19.76.1.2 pX

[CK\\_BYTE\\_PTR](#) pX

#### 19.76.1.3 ulXLen

[CK\\_ULONG](#) ulXLen

## 19.77 CK\_KIP\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

## Data Fields

- [CK\\_MECHANISM\\_PTR](#) pMechanism
- [CK\\_OBJECT\\_HANDLE](#) hKey
- [CK\\_BYTE\\_PTR](#) pSeed
- [CK\\_ULONG](#) ulSeedLen

### 19.77.1 Field Documentation

#### 19.77.1.1 hKey

[CK\\_OBJECT\\_HANDLE](#) hKey

#### 19.77.1.2 pMechanism

[CK\\_MECHANISM\\_PTR](#) pMechanism

#### 19.77.1.3 pSeed

[CK\\_BYTE\\_PTR](#) pSeed

#### 19.77.1.4 ulSeedLen

[CK\\_ULONG](#) ulSeedLen

## 19.78 CK\_MECHANISM Struct Reference

```
#include <pkcs11t.h>
```

## Data Fields

- [CK\\_MECHANISM\\_TYPE](#) mechanism
- [CK\\_VOID\\_PTR](#) pParameter
- [CK\\_ULONG](#) ulParameterLen

### 19.78.1 Field Documentation

#### 19.78.1.1 mechanism

[CK\\_MECHANISM\\_TYPE](#) mechanism

#### 19.78.1.2 pParameter

[CK\\_VOID\\_PTR](#) pParameter

#### 19.78.1.3 ulParameterLen

[CK\\_ULONG](#) ulParameterLen

## 19.79 CK\_MECHANISM\_INFO Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_ULONG](#) ulMinKeySize
- [CK\\_ULONG](#) ulMaxKeySize
- [CK\\_FLAGS](#) flags

### 19.79.1 Field Documentation

#### 19.79.1.1 flags

[CK\\_FLAGS](#) flags



### 19.79.1.2 ulMaxKeySize

`CK_ULONG` ulMaxKeySize

### 19.79.1.3 ulMinKeySize

`CK_ULONG` ulMinKeySize

## 19.80 CK\_OTP\_PARAM Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_OTP_PARAM_TYPE` type
- `CK_VOID_PTR` pValue
- `CK_ULONG` ulValueLen

### 19.80.1 Field Documentation

#### 19.80.1.1 pValue

`CK_VOID_PTR` pValue

#### 19.80.1.2 type

`CK_OTP_PARAM_TYPE` type

#### 19.80.1.3 ulValueLen

`CK_ULONG` ulValueLen

## 19.81 CK\_OTP\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_OTP\\_PARAM\\_PTR](#) pParams
- [CK\\_ULONG](#) ulCount

#### 19.81.1 Field Documentation

##### 19.81.1.1 pParams

[CK\\_OTP\\_PARAM\\_PTR](#) pParams

##### 19.81.1.2 ulCount

[CK\\_ULONG](#) ulCount

## 19.82 CK\_OTP\_SIGNATURE\_INFO Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_OTP\\_PARAM\\_PTR](#) pParams
- [CK\\_ULONG](#) ulCount

#### 19.82.1 Field Documentation

##### 19.82.1.1 pParams

[CK\\_OTP\\_PARAM\\_PTR](#) pParams

##### 19.82.1.2 ulCount

[CK\\_ULONG](#) ulCount

## 19.83 CK\_PBE\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_BYTE\\_PTR](#) pInitVector
- [CK\\_UTF8CHAR\\_PTR](#) pPassword
- [CK\\_ULONG](#) ulPasswordLen
- [CK\\_BYTE\\_PTR](#) pSalt
- [CK\\_ULONG](#) ulSaltLen
- [CK\\_ULONG](#) ulIteration

### 19.83.1 Field Documentation

#### 19.83.1.1 pInitVector

[CK\\_BYTE\\_PTR](#) pInitVector

#### 19.83.1.2 pPassword

[CK\\_UTF8CHAR\\_PTR](#) pPassword

#### 19.83.1.3 pSalt

[CK\\_BYTE\\_PTR](#) pSalt

#### 19.83.1.4 ulIteration

[CK\\_ULONG](#) ulIteration

#### 19.83.1.5 ulPasswordLen

[CK\\_ULONG](#) ulPasswordLen

### 19.83.1.6 ulSaltLen

`CK_ULONG ulSaltLen`

## 19.84 CK\_PKCS5\_PBKD2\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE saltSource`
- `CK_VOID_PTR pSaltSourceData`
- `CK_ULONG ulSaltSourceDataLen`
- `CK_ULONG iterations`
- `CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE prf`
- `CK_VOID_PTR pPrfData`
- `CK_ULONG ulPrfDataLen`
- `CK_UTF8CHAR_PTR pPassword`
- `CK_ULONG_PTR ulPasswordLen`

### 19.84.1 Field Documentation

#### 19.84.1.1 iterations

`CK_ULONG iterations`

#### 19.84.1.2 pPassword

`CK_UTF8CHAR_PTR pPassword`

#### 19.84.1.3 pPrfData

`CK_VOID_PTR pPrfData`

#### 19.84.1.4 prf

[CK\\_PKCS5\\_PBKD2\\_PSEUDO\\_RANDOM\\_FUNCTION\\_TYPE](#) prf

#### 19.84.1.5 pSaltSourceData

[CK\\_VOID\\_PTR](#) pSaltSourceData

#### 19.84.1.6 saltSource

[CK\\_PKCS5\\_PBKDF2\\_SALT\\_SOURCE\\_TYPE](#) saltSource

#### 19.84.1.7 ulPasswordLen

[CK\\_ULONG\\_PTR](#) ulPasswordLen

#### 19.84.1.8 ulPrfDataLen

[CK\\_ULONG](#) ulPrfDataLen

#### 19.84.1.9 ulSaltSourceDataLen

[CK\\_ULONG](#) ulSaltSourceDataLen

### 19.85 CK\_PKCS5\_PBKD2\_PARAMS2 Struct Reference

```
#include <pkcs11t.h>
```

#### Data Fields

- [CK\\_PKCS5\\_PBKDF2\\_SALT\\_SOURCE\\_TYPE](#) saltSource
- [CK\\_VOID\\_PTR](#) pSaltSourceData
- [CK\\_ULONG](#) ulSaltSourceDataLen
- [CK\\_ULONG](#) iterations
- [CK\\_PKCS5\\_PBKD2\\_PSEUDO\\_RANDOM\\_FUNCTION\\_TYPE](#) prf
- [CK\\_VOID\\_PTR](#) pPrfData
- [CK\\_ULONG](#) ulPrfDataLen
- [CK\\_UTF8CHAR\\_PTR](#) pPassword
- [CK\\_ULONG](#) ulPasswordLen

### 19.85.1 Field Documentation

#### 19.85.1.1 iterations

`CK_ULONG` iterations

#### 19.85.1.2 pPassword

`CK_UTF8CHAR_PTR` pPassword

#### 19.85.1.3 pPrfData

`CK_VOID_PTR` pPrfData

#### 19.85.1.4 prf

`CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE` prf

#### 19.85.1.5 pSaltSourceData

`CK_VOID_PTR` pSaltSourceData

#### 19.85.1.6 saltSource

`CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE` saltSource

#### 19.85.1.7 ulPasswordLen

`CK_ULONG` ulPasswordLen

### 19.85.1.8 ulPrfDataLen

`CK_ULONG ulPrfDataLen`

### 19.85.1.9 ulSaltSourceDataLen

`CK_ULONG ulSaltSourceDataLen`

## 19.86 CK\_RC2\_CBC\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_ULONG ulEffectiveBits`
- `CK_BYTE iv [8]`

### 19.86.1 Field Documentation

#### 19.86.1.1 iv

`CK_BYTE iv[8]`

#### 19.86.1.2 ulEffectiveBits

`CK_ULONG ulEffectiveBits`

## 19.87 CK\_RC2\_MAC\_GENERAL\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_ULONG ulEffectiveBits`
- `CK_ULONG ulMacLength`

### 19.87.1 Field Documentation

#### 19.87.1.1 ulEffectiveBits

`CK_ULONG` ulEffectiveBits

#### 19.87.1.2 ulMacLength

`CK_ULONG` ulMacLength

## 19.88 CK\_RC5\_CBC\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_ULONG` ulWordsize
- `CK_ULONG` ulRounds
- `CK_BYTE_PTR` pIv
- `CK_ULONG` ulIvLen

### 19.88.1 Field Documentation

#### 19.88.1.1 pIv

`CK_BYTE_PTR` pIv

#### 19.88.1.2 ulIvLen

`CK_ULONG` ulIvLen



### 19.88.1.3 ulRounds

`CK_ULONG` ulRounds

### 19.88.1.4 ulWordsize

`CK_ULONG` ulWordsize

## 19.89 CK\_RC5\_MAC\_GENERAL\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_ULONG` ulWordsize
- `CK_ULONG` ulRounds
- `CK_ULONG` ulMacLength

### 19.89.1 Field Documentation

#### 19.89.1.1 ulMacLength

`CK_ULONG` ulMacLength

#### 19.89.1.2 ulRounds

`CK_ULONG` ulRounds

#### 19.89.1.3 ulWordsize

`CK_ULONG` ulWordsize

## 19.90 CK\_RC5\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_ULONG](#) ulWordsize
- [CK\\_ULONG](#) ulRounds

#### 19.90.1 Field Documentation

##### 19.90.1.1 ulRounds

[CK\\_ULONG](#) ulRounds

##### 19.90.1.2 ulWordsize

[CK\\_ULONG](#) ulWordsize

## 19.91 CK\_RSA\_AES\_KEY\_WRAP\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_ULONG](#) ulAESKeyBits
- [CK\\_RSA\\_PKCS\\_OAEP\\_PARAMS\\_PTR](#) pOAEPParams

#### 19.91.1 Field Documentation

##### 19.91.1.1 pOAEPParams

[CK\\_RSA\\_PKCS\\_OAEP\\_PARAMS\\_PTR](#) pOAEPParams

##### 19.91.1.2 ulAESKeyBits

[CK\\_ULONG](#) ulAESKeyBits

## 19.92 CK\_RSA\_PKCS\_OAEP\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_MECHANISM\\_TYPE](#) hashAlg
- [CK\\_RSA\\_PKCS\\_MGF\\_TYPE](#) mgf
- [CK\\_RSA\\_PKCS\\_OAEP\\_SOURCE\\_TYPE](#) source
- [CK\\_VOID\\_PTR](#) pSourceData
- [CK\\_ULONG](#) ulSourceDataLen

### 19.92.1 Field Documentation

#### 19.92.1.1 hashAlg

[CK\\_MECHANISM\\_TYPE](#) hashAlg

#### 19.92.1.2 mgf

[CK\\_RSA\\_PKCS\\_MGF\\_TYPE](#) mgf

#### 19.92.1.3 pSourceData

[CK\\_VOID\\_PTR](#) pSourceData

#### 19.92.1.4 source

[CK\\_RSA\\_PKCS\\_OAEP\\_SOURCE\\_TYPE](#) source

#### 19.92.1.5 ulSourceDataLen

[CK\\_ULONG](#) ulSourceDataLen

## 19.93 CK\_RSA\_PKCS\_PSS\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_MECHANISM\\_TYPE](#) hashAlg
- [CK\\_RSA\\_PKCS\\_MGF\\_TYPE](#) mgf
- [CK\\_ULONG](#) sLen

### 19.93.1 Field Documentation

#### 19.93.1.1 hashAlg

[CK\\_MECHANISM\\_TYPE](#) hashAlg

#### 19.93.1.2 mgf

[CK\\_RSA\\_PKCS\\_MGF\\_TYPE](#) mgf

#### 19.93.1.3 sLen

[CK\\_ULONG](#) sLen

## 19.94 CK\_SEED\_CBC\_ENCRYPT\_DATA\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_BYTE](#) iv [16]
- [CK\\_BYTE\\_PTR](#) pData
- [CK\\_ULONG](#) length

### 19.94.1 Field Documentation

#### 19.94.1.1 iv

`CK_BYTE iv[16]`

#### 19.94.1.2 length

`CK_ULONG length`

#### 19.94.1.3 pData

`CK_BYTE_PTR pData`

### 19.95 CK\_SESSION\_INFO Struct Reference

```
#include <pkcs11t.h>
```

#### Data Fields

- `CK_SLOT_ID slotID`
- `CK_STATE state`
- `CK_FLAGS flags`
- `CK_ULONG ulDeviceError`

#### 19.95.1 Field Documentation

##### 19.95.1.1 flags

`CK_FLAGS flags`

##### 19.95.1.2 slotID

`CK_SLOT_ID slotID`

### 19.95.1.3 state

`CK_STATE` state

### 19.95.1.4 ulDeviceError

`CK_ULONG` ulDeviceError

## 19.96 CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_ULONG` ulPasswordLen
- `CK_BYTE_PTR` pPassword
- `CK_ULONG` ulPublicDataLen
- `CK_BYTE_PTR` pPublicData
- `CK_ULONG` ulPAndGLen
- `CK_ULONG` ulQLen
- `CK_ULONG` ulRandomLen
- `CK_BYTE_PTR` pRandomA
- `CK_BYTE_PTR` pPrimeP
- `CK_BYTE_PTR` pBaseG
- `CK_BYTE_PTR` pSubprimeQ

### 19.96.1 Field Documentation

#### 19.96.1.1 pBaseG

`CK_BYTE_PTR` pBaseG

#### 19.96.1.2 pPassword

`CK_BYTE_PTR` pPassword

**19.96.1.3 pPrimeP**

`CK_BYTE_PTR` pPrimeP

**19.96.1.4 pPublicData**

`CK_BYTE_PTR` pPublicData

**19.96.1.5 pRandomA**

`CK_BYTE_PTR` pRandomA

**19.96.1.6 pSubprimeQ**

`CK_BYTE_PTR` pSubprimeQ

**19.96.1.7 ulPAndGLen**

`CK_ULONG` ulPAndGLen

**19.96.1.8 ulPasswordLen**

`CK_ULONG` ulPasswordLen

**19.96.1.9 ulPublicDataLen**

`CK_ULONG` ulPublicDataLen

**19.96.1.10 ulQLen**

`CK_ULONG` ulQLen

### 19.96.1.11 ulRandomLen

[CK\\_ULONG](#) ulRandomLen

## 19.97 CK\_SKIPJACK\_RELAYX\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_ULONG](#) ulOldWrappedXLen
- [CK\\_BYTE\\_PTR](#) pOldWrappedX
- [CK\\_ULONG](#) ulOldPasswordLen
- [CK\\_BYTE\\_PTR](#) pOldPassword
- [CK\\_ULONG](#) ulOldPublicDataLen
- [CK\\_BYTE\\_PTR](#) pOldPublicData
- [CK\\_ULONG](#) ulOldRandomLen
- [CK\\_BYTE\\_PTR](#) pOldRandomA
- [CK\\_ULONG](#) ulNewPasswordLen
- [CK\\_BYTE\\_PTR](#) pNewPassword
- [CK\\_ULONG](#) ulNewPublicDataLen
- [CK\\_BYTE\\_PTR](#) pNewPublicData
- [CK\\_ULONG](#) ulNewRandomLen
- [CK\\_BYTE\\_PTR](#) pNewRandomA

### 19.97.1 Field Documentation

#### 19.97.1.1 pNewPassword

[CK\\_BYTE\\_PTR](#) pNewPassword

#### 19.97.1.2 pNewPublicData

[CK\\_BYTE\\_PTR](#) pNewPublicData

#### 19.97.1.3 pNewRandomA

[CK\\_BYTE\\_PTR](#) pNewRandomA



**19.97.1.4 pOldPassword**

`CK_BYTE_PTR` pOldPassword

**19.97.1.5 pOldPublicData**

`CK_BYTE_PTR` pOldPublicData

**19.97.1.6 pOldRandomA**

`CK_BYTE_PTR` pOldRandomA

**19.97.1.7 pOldWrappedX**

`CK_BYTE_PTR` pOldWrappedX

**19.97.1.8 ulNewPasswordLen**

`CK_ULONG` ulNewPasswordLen

**19.97.1.9 ulNewPublicDataLen**

`CK_ULONG` ulNewPublicDataLen

**19.97.1.10 ulNewRandomLen**

`CK_ULONG` ulNewRandomLen

**19.97.1.11 ulOldPasswordLen**

`CK_ULONG` ulOldPasswordLen

### 19.97.1.12 ulOldPublicDataLen

`CK_ULONG` ulOldPublicDataLen

### 19.97.1.13 ulOldRandomLen

`CK_ULONG` ulOldRandomLen

### 19.97.1.14 ulOldWrappedXLen

`CK_ULONG` ulOldWrappedXLen

## 19.98 CK\_SLOT\_INFO Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_UTF8CHAR` slotDescription [64]
- `CK_UTF8CHAR` manufacturerID [32]
- `CK_FLAGS` flags
- `CK_VERSION` hardwareVersion
- `CK_VERSION` firmwareVersion

### 19.98.1 Field Documentation

#### 19.98.1.1 firmwareVersion

`CK_VERSION` firmwareVersion

#### 19.98.1.2 flags

`CK_FLAGS` flags

### 19.98.1.3 hardwareVersion

`CK_VERSION` hardwareVersion

### 19.98.1.4 manufacturerID

`CK_UTF8CHAR` manufacturerID[32]

### 19.98.1.5 slotDescription

`CK_UTF8CHAR` slotDescription[64]

## 19.99 CK\_SSL3\_KEY\_MAT\_OUT Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_OBJECT_HANDLE` hClientMacSecret
- `CK_OBJECT_HANDLE` hServerMacSecret
- `CK_OBJECT_HANDLE` hClientKey
- `CK_OBJECT_HANDLE` hServerKey
- `CK_BYTE_PTR` pIVClient
- `CK_BYTE_PTR` pIVServer

### 19.99.1 Field Documentation

#### 19.99.1.1 hClientKey

`CK_OBJECT_HANDLE` hClientKey

#### 19.99.1.2 hClientMacSecret

`CK_OBJECT_HANDLE` hClientMacSecret

### 19.99.1.3 hServerKey

[CK\\_OBJECT\\_HANDLE](#) hServerKey

### 19.99.1.4 hServerMacSecret

[CK\\_OBJECT\\_HANDLE](#) hServerMacSecret

### 19.99.1.5 pIVClient

[CK\\_BYTE\\_PTR](#) pIVClient

### 19.99.1.6 pIVServer

[CK\\_BYTE\\_PTR](#) pIVServer

## 19.100 CK\_SSL3\_KEY\_MAT\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_ULONG](#) ulMacSizeInBits
- [CK\\_ULONG](#) ulKeySizeInBits
- [CK\\_ULONG](#) ulIVSizeInBits
- [CK\\_BBOOL](#) blsExport
- [CK\\_SSL3\\_RANDOM\\_DATA](#) RandomInfo
- [CK\\_SSL3\\_KEY\\_MAT\\_OUT\\_PTR](#) pReturnedKeyMaterial

### 19.100.1 Field Documentation

#### 19.100.1.1 blsExport

[CK\\_BBOOL](#) blsExport

#### 19.100.1.2 pReturnedKeyMaterial

`CK_SSL3_KEY_MAT_OUT_PTR` pReturnedKeyMaterial

#### 19.100.1.3 RandomInfo

`CK_SSL3_RANDOM_DATA` RandomInfo

#### 19.100.1.4 ulIVSizeInBits

`CK_ULONG` ulIVSizeInBits

#### 19.100.1.5 ulKeySizeInBits

`CK_ULONG` ulKeySizeInBits

#### 19.100.1.6 ulMacSizeInBits

`CK_ULONG` ulMacSizeInBits

### 19.101 CK\_SSL3\_MASTER\_KEY\_DERIVE\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

#### Data Fields

- `CK_SSL3_RANDOM_DATA` RandomInfo
- `CK_VERSION_PTR` pVersion

#### 19.101.1 Field Documentation

### 19.101.1.1 pVersion

[CK\\_VERSION\\_PTR](#) pVersion

### 19.101.1.2 RandomInfo

[CK\\_SSL3\\_RANDOM\\_DATA](#) RandomInfo

## 19.102 CK\_SSL3\_RANDOM\_DATA Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_BYTE\\_PTR](#) pClientRandom
- [CK\\_ULONG](#) ulClientRandomLen
- [CK\\_BYTE\\_PTR](#) pServerRandom
- [CK\\_ULONG](#) ulServerRandomLen

### 19.102.1 Field Documentation

#### 19.102.1.1 pClientRandom

[CK\\_BYTE\\_PTR](#) pClientRandom

#### 19.102.1.2 pServerRandom

[CK\\_BYTE\\_PTR](#) pServerRandom

#### 19.102.1.3 ulClientRandomLen

[CK\\_ULONG](#) ulClientRandomLen

#### 19.102.1.4 ulServerRandomLen

`CK_ULONG ulServerRandomLen`

### 19.103 CK\_TLS12\_KEY\_MAT\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

#### Data Fields

- `CK_ULONG ulMacSizeInBits`
- `CK_ULONG ulKeySizeInBits`
- `CK_ULONG ulIVSizeInBits`
- `CK_BBOOL blsExport`
- `CK_SSL3_RANDOM_DATA RandomInfo`
- `CK_SSL3_KEY_MAT_OUT_PTR pReturnedKeyMaterial`
- `CK_MECHANISM_TYPE prfHashMechanism`

#### 19.103.1 Field Documentation

##### 19.103.1.1 blsExport

`CK_BBOOL blsExport`

##### 19.103.1.2 pReturnedKeyMaterial

`CK_SSL3_KEY_MAT_OUT_PTR pReturnedKeyMaterial`

##### 19.103.1.3 prfHashMechanism

`CK_MECHANISM_TYPE prfHashMechanism`

##### 19.103.1.4 RandomInfo

`CK_SSL3_RANDOM_DATA RandomInfo`

### 19.103.1.5 ulIVSizeInBits

[CK\\_ULONG](#) ulIVSizeInBits

### 19.103.1.6 ulKeySizeInBits

[CK\\_ULONG](#) ulKeySizeInBits

### 19.103.1.7 ulMacSizeInBits

[CK\\_ULONG](#) ulMacSizeInBits

## 19.104 CK\_TLS12\_MASTER\_KEY\_DERIVE\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_SSL3\\_RANDOM\\_DATA](#) RandomInfo
- [CK\\_VERSION\\_PTR](#) pVersion
- [CK\\_MECHANISM\\_TYPE](#) prfHashMechanism

### 19.104.1 Field Documentation

#### 19.104.1.1 prfHashMechanism

[CK\\_MECHANISM\\_TYPE](#) prfHashMechanism

#### 19.104.1.2 pVersion

[CK\\_VERSION\\_PTR](#) pVersion



### 19.104.1.3 RandomInfo

[CK\\_SSL3\\_RANDOM\\_DATA](#) RandomInfo

## 19.105 CK\_TLS\_KDF\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_MECHANISM\\_TYPE](#) prfMechanism
- [CK\\_BYTE\\_PTR](#) pLabel
- [CK\\_ULONG](#) ulLabelLength
- [CK\\_SSL3\\_RANDOM\\_DATA](#) RandomInfo
- [CK\\_BYTE\\_PTR](#) pContextData
- [CK\\_ULONG](#) ulContextDataLength

### 19.105.1 Field Documentation

#### 19.105.1.1 pContextData

[CK\\_BYTE\\_PTR](#) pContextData

#### 19.105.1.2 pLabel

[CK\\_BYTE\\_PTR](#) pLabel

#### 19.105.1.3 prfMechanism

[CK\\_MECHANISM\\_TYPE](#) prfMechanism

#### 19.105.1.4 RandomInfo

[CK\\_SSL3\\_RANDOM\\_DATA](#) RandomInfo

## 19.106 CK\_TLS\_MAC\_PARAMS Struct Reference

---

### 19.105.1.5 ulContextDataLength

`CK_ULONG` ulContextDataLength

### 19.105.1.6 ulLabelLength

`CK_ULONG` ulLabelLength

## 19.106 CK\_TLS\_MAC\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- `CK_MECHANISM_TYPE` prfHashMechanism
- `CK_ULONG` ulMacLength
- `CK_ULONG` ulServerOrClient

### 19.106.1 Field Documentation

#### 19.106.1.1 prfHashMechanism

`CK_MECHANISM_TYPE` prfHashMechanism

#### 19.106.1.2 ulMacLength

`CK_ULONG` ulMacLength

#### 19.106.1.3 ulServerOrClient

`CK_ULONG` ulServerOrClient

## 19.107 CK\_TLS\_PRF\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

## Data Fields

- [CK\\_BYTE\\_PTR](#) pSeed
- [CK\\_ULONG](#) ulSeedLen
- [CK\\_BYTE\\_PTR](#) pLabel
- [CK\\_ULONG](#) ulLabelLen
- [CK\\_BYTE\\_PTR](#) pOutput
- [CK\\_ULONG\\_PTR](#) pulOutputLen

### 19.107.1 Field Documentation

#### 19.107.1.1 pLabel

[CK\\_BYTE\\_PTR](#) pLabel

#### 19.107.1.2 pOutput

[CK\\_BYTE\\_PTR](#) pOutput

#### 19.107.1.3 pSeed

[CK\\_BYTE\\_PTR](#) pSeed

#### 19.107.1.4 pulOutputLen

[CK\\_ULONG\\_PTR](#) pulOutputLen

#### 19.107.1.5 ulLabelLen

[CK\\_ULONG](#) ulLabelLen

#### 19.107.1.6 ulSeedLen

[CK\\_ULONG](#) ulSeedLen

## 19.108 CK\_TOKEN\_INFO Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_UTF8CHAR](#) label [32]
- [CK\\_UTF8CHAR](#) manufacturerID [32]
- [CK\\_UTF8CHAR](#) model [16]
- [CK\\_CHAR](#) serialNumber [16]
- [CK\\_FLAGS](#) flags
- [CK\\_ULONG](#) ulMaxSessionCount
- [CK\\_ULONG](#) ulSessionCount
- [CK\\_ULONG](#) ulMaxRwSessionCount
- [CK\\_ULONG](#) ulRwSessionCount
- [CK\\_ULONG](#) ulMaxPinLen
- [CK\\_ULONG](#) ulMinPinLen
- [CK\\_ULONG](#) ulTotalPublicMemory
- [CK\\_ULONG](#) ulFreePublicMemory
- [CK\\_ULONG](#) ulTotalPrivateMemory
- [CK\\_ULONG](#) ulFreePrivateMemory
- [CK\\_VERSION](#) hardwareVersion
- [CK\\_VERSION](#) firmwareVersion
- [CK\\_CHAR](#) utcTime [16]

### 19.108.1 Field Documentation

#### 19.108.1.1 firmwareVersion

[CK\\_VERSION](#) firmwareVersion

#### 19.108.1.2 flags

[CK\\_FLAGS](#) flags

#### 19.108.1.3 hardwareVersion

[CK\\_VERSION](#) hardwareVersion

**19.108.1.4 label**

`CK_UTF8CHAR label[32]`

**19.108.1.5 manufacturerID**

`CK_UTF8CHAR manufacturerID[32]`

**19.108.1.6 model**

`CK_UTF8CHAR model[16]`

**19.108.1.7 serialNumber**

`CK_CHAR serialNumber[16]`

**19.108.1.8 ulFreePrivateMemory**

`CK_ULONG ulFreePrivateMemory`

**19.108.1.9 ulFreePublicMemory**

`CK_ULONG ulFreePublicMemory`

**19.108.1.10 ulMaxPinLen**

`CK_ULONG ulMaxPinLen`

**19.108.1.11 ulMaxRwSessionCount**

`CK_ULONG ulMaxRwSessionCount`

### 19.108.1.12 ulMaxSessionCount

`CK_ULONG` ulMaxSessionCount

### 19.108.1.13 ulMinPinLen

`CK_ULONG` ulMinPinLen

### 19.108.1.14 ulRwSessionCount

`CK_ULONG` ulRwSessionCount

### 19.108.1.15 ulSessionCount

`CK_ULONG` ulSessionCount

### 19.108.1.16 ulTotalPrivateMemory

`CK_ULONG` ulTotalPrivateMemory

### 19.108.1.17 ulTotalPublicMemory

`CK_ULONG` ulTotalPublicMemory

### 19.108.1.18 utcTime

`CK_CHAR` utcTime[16]

## 19.109 CK\_VERSION Struct Reference

```
#include <pkcs11t.h>
```

## Data Fields

- [CK\\_BYTE](#) major
- [CK\\_BYTE](#) minor

### 19.109.1 Field Documentation

#### 19.109.1.1 major

[CK\\_BYTE](#) major

#### 19.109.1.2 minor

[CK\\_BYTE](#) minor

## 19.110 CK\_WTLS\_KEY\_MAT\_OUT Struct Reference

```
#include <pkcs11t.h>
```

## Data Fields

- [CK\\_OBJECT\\_HANDLE](#) hMacSecret
- [CK\\_OBJECT\\_HANDLE](#) hKey
- [CK\\_BYTE\\_PTR](#) pIV

### 19.110.1 Field Documentation

#### 19.110.1.1 hKey

[CK\\_OBJECT\\_HANDLE](#) hKey

#### 19.110.1.2 hMacSecret

[CK\\_OBJECT\\_HANDLE](#) hMacSecret

### 19.110.1.3 pIV

[CK\\_BYTE\\_PTR](#) pIV

## 19.111 CK\_WTLS\_KEY\_MAT\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_MECHANISM\\_TYPE](#) DigestMechanism
- [CK\\_ULONG](#) ulMacSizeInBits
- [CK\\_ULONG](#) ulKeySizeInBits
- [CK\\_ULONG](#) ulIVSizeInBits
- [CK\\_ULONG](#) ulSequenceNumber
- [CK\\_BBOOL](#) blsExport
- [CK\\_WTLS\\_RANDOM\\_DATA](#) RandomInfo
- [CK\\_WTLS\\_KEY\\_MAT\\_OUT\\_PTR](#) pReturnedKeyMaterial

### 19.111.1 Field Documentation

#### 19.111.1.1 blsExport

[CK\\_BBOOL](#) blsExport

#### 19.111.1.2 DigestMechanism

[CK\\_MECHANISM\\_TYPE](#) DigestMechanism

#### 19.111.1.3 pReturnedKeyMaterial

[CK\\_WTLS\\_KEY\\_MAT\\_OUT\\_PTR](#) pReturnedKeyMaterial

#### 19.111.1.4 RandomInfo

[CK\\_WTLS\\_RANDOM\\_DATA](#) RandomInfo



#### 19.111.1.5 ulIVSizeInBits

`CK_ULONG ulIVSizeInBits`

#### 19.111.1.6 ulKeySizeInBits

`CK_ULONG ulKeySizeInBits`

#### 19.111.1.7 ulMacSizeInBits

`CK_ULONG ulMacSizeInBits`

#### 19.111.1.8 ulSequenceNumber

`CK_ULONG ulSequenceNumber`

### 19.112 CK\_WTLS\_MASTER\_KEY\_DERIVE\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

#### Data Fields

- `CK_MECHANISM_TYPE` DigestMechanism
- `CK_WTLS_RANDOM_DATA` RandomInfo
- `CK_BYTE_PTR` pVersion

#### 19.112.1 Field Documentation

##### 19.112.1.1 DigestMechanism

`CK_MECHANISM_TYPE` DigestMechanism

### 19.112.1.2 pVersion

[CK\\_BYTE\\_PTR](#) pVersion

### 19.112.1.3 RandomInfo

[CK\\_WTLS\\_RANDOM\\_DATA](#) RandomInfo

## 19.113 CK\_WTLS\_PRF\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_MECHANISM\\_TYPE](#) DigestMechanism
- [CK\\_BYTE\\_PTR](#) pSeed
- [CK\\_ULONG](#) ulSeedLen
- [CK\\_BYTE\\_PTR](#) pLabel
- [CK\\_ULONG](#) ulLabelLen
- [CK\\_BYTE\\_PTR](#) pOutput
- [CK\\_ULONG\\_PTR](#) pulOutputLen

### 19.113.1 Field Documentation

#### 19.113.1.1 DigestMechanism

[CK\\_MECHANISM\\_TYPE](#) DigestMechanism

#### 19.113.1.2 pLabel

[CK\\_BYTE\\_PTR](#) pLabel

#### 19.113.1.3 pOutput

[CK\\_BYTE\\_PTR](#) pOutput

#### 19.113.1.4 pSeed

[CK\\_BYTE\\_PTR](#) pSeed

#### 19.113.1.5 pulOutputLen

[CK\\_ULONG\\_PTR](#) pulOutputLen

#### 19.113.1.6 ulLabelLen

[CK\\_ULONG](#) ulLabelLen

#### 19.113.1.7 ulSeedLen

[CK\\_ULONG](#) ulSeedLen

### 19.114 CK\_WTLS\_RANDOM\_DATA Struct Reference

```
#include <pkcs11t.h>
```

#### Data Fields

- [CK\\_BYTE\\_PTR](#) pClientRandom
- [CK\\_ULONG](#) ulClientRandomLen
- [CK\\_BYTE\\_PTR](#) pServerRandom
- [CK\\_ULONG](#) ulServerRandomLen

#### 19.114.1 Field Documentation

##### 19.114.1.1 pClientRandom

[CK\\_BYTE\\_PTR](#) pClientRandom

### 19.114.1.2 pServerRandom

[CK\\_BYTE\\_PTR](#) pServerRandom

### 19.114.1.3 ulClientRandomLen

[CK\\_ULONG](#) ulClientRandomLen

### 19.114.1.4 ulServerRandomLen

[CK\\_ULONG](#) ulServerRandomLen

## 19.115 CK\_X9\_42\_DH1\_DERIVE\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_X9\\_42\\_DH\\_KDF\\_TYPE](#) kdf
- [CK\\_ULONG](#) ulOtherInfoLen
- [CK\\_BYTE\\_PTR](#) pOtherInfo
- [CK\\_ULONG](#) ulPublicDataLen
- [CK\\_BYTE\\_PTR](#) pPublicData

### 19.115.1 Field Documentation

#### 19.115.1.1 kdf

[CK\\_X9\\_42\\_DH\\_KDF\\_TYPE](#) kdf

#### 19.115.1.2 pOtherInfo

[CK\\_BYTE\\_PTR](#) pOtherInfo

#### 19.115.1.3 pPublicData

`CK_BYTE_PTR` pPublicData

#### 19.115.1.4 ulOtherInfoLen

`CK_ULONG` ulOtherInfoLen

#### 19.115.1.5 ulPublicDataLen

`CK_ULONG` ulPublicDataLen

### 19.116 CK\_X9\_42\_DH2\_DERIVE\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

#### Data Fields

- `CK_X9_42_DH_KDF_TYPE` kdf
- `CK_ULONG` ulOtherInfoLen
- `CK_BYTE_PTR` pOtherInfo
- `CK_ULONG` ulPublicDataLen
- `CK_BYTE_PTR` pPublicData
- `CK_ULONG` ulPrivateDataLen
- `CK_OBJECT_HANDLE` hPrivateData
- `CK_ULONG` ulPublicDataLen2
- `CK_BYTE_PTR` pPublicData2

#### 19.116.1 Field Documentation

##### 19.116.1.1 hPrivateData

`CK_OBJECT_HANDLE` hPrivateData

### 19.116.1.2 kdf

[CK\\_X9\\_42\\_DH\\_KDF\\_TYPE](#) kdf

### 19.116.1.3 pOtherInfo

[CK\\_BYTE\\_PTR](#) pOtherInfo

### 19.116.1.4 pPublicData

[CK\\_BYTE\\_PTR](#) pPublicData

### 19.116.1.5 pPublicData2

[CK\\_BYTE\\_PTR](#) pPublicData2

### 19.116.1.6 ulOtherInfoLen

[CK\\_ULONG](#) ulOtherInfoLen

### 19.116.1.7 ulPrivateDataLen

[CK\\_ULONG](#) ulPrivateDataLen

### 19.116.1.8 ulPublicDataLen

[CK\\_ULONG](#) ulPublicDataLen

### 19.116.1.9 ulPublicDataLen2

[CK\\_ULONG](#) ulPublicDataLen2

## 19.117 CK\_X9\_42\_MQV\_DERIVE\_PARAMS Struct Reference

```
#include <pkcs11t.h>
```

### Data Fields

- [CK\\_X9\\_42\\_DH\\_KDF\\_TYPE](#) kdf
- [CK\\_ULONG](#) ulOtherInfoLen
- [CK\\_BYTE\\_PTR](#) pOtherInfo
- [CK\\_ULONG](#) ulPublicDataLen
- [CK\\_BYTE\\_PTR](#) pPublicData
- [CK\\_ULONG](#) ulPrivateDataLen
- [CK\\_OBJECT\\_HANDLE](#) hPrivateData
- [CK\\_ULONG](#) ulPublicDataLen2
- [CK\\_BYTE\\_PTR](#) pPublicData2
- [CK\\_OBJECT\\_HANDLE](#) publicKey

### 19.117.1 Field Documentation

#### 19.117.1.1 hPrivateData

[CK\\_OBJECT\\_HANDLE](#) hPrivateData

#### 19.117.1.2 kdf

[CK\\_X9\\_42\\_DH\\_KDF\\_TYPE](#) kdf

#### 19.117.1.3 pOtherInfo

[CK\\_BYTE\\_PTR](#) pOtherInfo

#### 19.117.1.4 pPublicData

[CK\\_BYTE\\_PTR](#) pPublicData

### 19.117.1.5 pPublicData2

[CK\\_BYTE\\_PTR](#) pPublicData2

### 19.117.1.6 publicKey

[CK\\_OBJECT\\_HANDLE](#) publicKey

### 19.117.1.7 ulOtherInfoLen

[CK\\_ULONG](#) ulOtherInfoLen

### 19.117.1.8 ulPrivateDataLen

[CK\\_ULONG](#) ulPrivateDataLen

### 19.117.1.9 ulPublicDataLen

[CK\\_ULONG](#) ulPublicDataLen

### 19.117.1.10 ulPublicDataLen2

[CK\\_ULONG](#) ulPublicDataLen2

## 19.118 CL\_HashContext Struct Reference

```
#include <sha1_routines.h>
```

### Data Fields

- [uint32\\_t](#) [h](#) [20/4]
- [uint32\\_t](#) [buf](#) [64/4]
- [uint32\\_t](#) [byteCount](#)
- [uint32\\_t](#) [byteCountHi](#)



## 19.118.1 Field Documentation

### 19.118.1.1 buf

```
uint32_t buf[64/4]
```

### 19.118.1.2 byteCount

```
uint32_t byteCount
```

### 19.118.1.3 byteCountHi

```
uint32_t byteCountHi
```

### 19.118.1.4 h

```
uint32_t h[20/4]
```

## 19.119 hid\_device Struct Reference

```
#include <hal_linux_kit_hid.h>
```

### Data Fields

- FILE \* [read\\_handle](#)
- FILE \* [write\\_handle](#)  
*The kit USB read file handle.*
- HANDLE [read\\_handle](#)
- HANDLE [write\\_handle](#)  
*The kit USB read file handle.*

## 19.119.1 Field Documentation

### 19.119.1.1 read\_handle [1/2]

FILE\* read\_handle

### 19.119.1.2 read\_handle [2/2]

HANDLE read\_handle

### 19.119.1.3 write\_handle [1/2]

FILE\* write\_handle

The kit USB read file handle.

### 19.119.1.4 write\_handle [2/2]

HANDLE write\_handle

The kit USB read file handle.

## 19.120 hw\_sha256\_ctx Struct Reference

### Data Fields

- uint32\_t [total\\_msg\\_size](#)  
*Total number of message bytes processed.*
- uint32\_t [block\\_size](#)  
*Number of bytes in current block.*
- uint8\_t [block](#) [[ATCA\\_SHA256\\_BLOCK\\_SIZE](#) \*2]  
*Unprocessed message storage.*

### 19.120.1 Field Documentation

#### 19.120.1.1 block

uint8\_t block[[ATCA\\_SHA256\\_BLOCK\\_SIZE](#) \*2]

Unprocessed message storage.

### 19.120.1.2 `block_size`

```
uint32_t block_size
```

Number of bytes in current block.

### 19.120.1.3 `total_msg_size`

```
uint32_t total_msg_size
```

Total number of message bytes processed.

## 19.121 `i2c_sam0_instance` Struct Reference

```
#include <hal_sam0_i2c_asf.h>
```

### Data Fields

- struct `i2c_master_module` \* [i2c\\_instance](#)
- [sam0\\_change\\_baudrate](#) `change_baudrate`

### 19.121.1 Field Documentation

#### 19.121.1.1 `change_baudrate`

```
sam0\_change\_baudrate change_baudrate
```

#### 19.121.1.2 `i2c_instance`

```
struct i2c_master_module* i2c_instance
```

## 19.122 `i2c_sam_instance` Struct Reference

```
#include <hal_sam_i2c_asf.h>
```

### Data Fields

- [Twi \\* i2c\\_instance](#)
- [sam\\_change\\_baudrate change\\_baudrate](#)

#### 19.122.1 Field Documentation

##### 19.122.1.1 change\_baudrate

[sam\\_change\\_baudrate](#) change\_baudrate

##### 19.122.1.2 i2c\_instance

`Twi* i2c_instance`

## 19.123 i2c\_start\_instance Struct Reference

```
#include <hal_i2c_start.h>
```

### Data Fields

- `struct i2c_m_sync_desc * i2c\_descriptor`
- [start\\_change\\_baudrate change\\_baudrate](#)

#### 19.123.1 Field Documentation

##### 19.123.1.1 change\_baudrate

[start\\_change\\_baudrate](#) change\_baudrate

##### 19.123.1.2 i2c\_descriptor

```
struct i2c_m_sync_desc* i2c_descriptor
```

## 19.124 memory\_parameters Struct Reference

```
#include <secure_boot_memory.h>
```

### Data Fields

- uint32\_t [start\\_address](#)
- uint32\_t [memory\\_size](#)
- uint32\_t [version\\_info](#)
- uint8\_t [reserved](#) [52]
- uint8\_t [signature](#) [[ATCA\\_SIG\\_SIZE](#)]

### 19.124.1 Field Documentation

#### 19.124.1.1 memory\_size

```
uint32_t memory_size
```

#### 19.124.1.2 reserved

```
uint8_t reserved[52]
```

#### 19.124.1.3 signature

```
uint8_t signature[ATCA\_SIG\_SIZE]
```

#### 19.124.1.4 start\_address

```
uint32_t start_address
```

#### 19.124.1.5 version\_info

```
uint32_t version_info
```

## 19.125 secure\_boot\_config\_bits Struct Reference

```
#include <secure_boot.h>
```

### Data Fields

- uint16\_t [secure\\_boot\\_mode](#): 2
- uint16\_t [secure\\_boot\\_reserved1](#): 1
- uint16\_t [secure\\_boot\\_persistent\\_enable](#): 1
- uint16\_t [secure\\_boot\\_rand\\_nonce](#): 1
- uint16\_t [secure\\_boot\\_reserved2](#): 3
- uint16\_t [secure\\_boot\\_sig\\_dig](#): 4
- uint16\_t [secure\\_boot\\_pub\\_key](#): 4

### 19.125.1 Field Documentation

#### 19.125.1.1 secure\_boot\_mode

```
uint16_t secure_boot_mode
```

#### 19.125.1.2 secure\_boot\_persistent\_enable

```
uint16_t secure_boot_persistent_enable
```

#### 19.125.1.3 secure\_boot\_pub\_key

```
uint16_t secure_boot_pub_key
```

#### 19.125.1.4 secure\_boot\_rand\_nonce

```
uint16_t secure_boot_rand_nonce
```

#### 19.125.1.5 secure\_boot\_reserved1

```
uint16_t secure_boot_reserved1
```

#### 19.125.1.6 secure\_boot\_reserved2

```
uint16_t secure_boot_reserved2
```

#### 19.125.1.7 secure\_boot\_sig\_dig

```
uint16_t secure_boot_sig_dig
```

### 19.126 secure\_boot\_parameters Struct Reference

```
#include <secure_boot.h>
```

#### Data Fields

- [memory\\_parameters](#) [memory\\_params](#)
- [atcac\\_sha2\\_256\\_ctx](#) [s\\_sha\\_context](#)
- [uint8\\_t](#) [app\\_digest](#) [[ATCA\\_SHA\\_DIGEST\\_SIZE](#)]

#### 19.126.1 Field Documentation

##### 19.126.1.1 app\_digest

```
uint8_t app_digest [ATCA\_SHA\_DIGEST\_SIZE]
```

##### 19.126.1.2 memory\_params

```
memory\_parameters memory_params
```

##### 19.126.1.3 s\_sha\_context

```
atcac\_sha2\_256\_ctx s_sha_context
```

### 19.127 sw\_sha256\_ctx Struct Reference

```
#include <sha2_routines.h>
```

### Data Fields

- uint32\_t [total\\_msg\\_size](#)  
*Total number of message bytes processed.*
- uint32\_t [block\\_size](#)  
*Number of bytes in current block.*
- uint8\_t [block](#) [(64) \*2]  
*Unprocessed message storage.*
- uint32\_t [hash](#) [8]  
*Hash state.*

### 19.127.1 Field Documentation

#### 19.127.1.1 block

```
uint8_t block[(64) *2]
```

Unprocessed message storage.

#### 19.127.1.2 block\_size

```
uint32_t block_size
```

Number of bytes in current block.

#### 19.127.1.3 hash

```
uint32_t hash[8]
```

Hash state.

#### 19.127.1.4 total\_msg\_size

```
uint32_t total_msg_size
```

Total number of message bytes processed.



## 19.128 tng\_cert\_map\_element Struct Reference

### Data Fields

- const char \* [otpcode](#)
- const [atccert\\_def\\_t](#) \* [cert\\_def](#)

### 19.128.1 Field Documentation

#### 19.128.1.1 cert\_def

```
const atccert\_def\_t* cert_def
```

#### 19.128.1.2 otpcode

```
const char* otpcode
```

## Chapter 20

# File Documentation

### 20.1 api\_206a.c File Reference

Provides APIs to use with ATSHA206A device.

```
#include <stdlib.h>
#include <stdio.h>
#include "cryptoauthlib.h"
#include "api_206a.h"
```

#### Functions

- [ATCA\\_STATUS sha206a\\_diversify\\_parent\\_key](#) (uint8\_t \*parent\_key, uint8\_t \*diversified\_key)  
*Computes the diversified key based on the parent key provided and device serial number.*
- [ATCA\\_STATUS sha206a\\_generate\\_derive\\_key](#) (uint8\_t \*parent\_key, uint8\_t \*derived\_key, uint8\_t param1, uint16\_t param2)  
*Generates the derived key based on the parent key and other parameters provided.*
- [ATCA\\_STATUS sha206a\\_generate\\_challenge\\_response\\_pair](#) (uint8\_t \*key, uint8\_t \*challenge, uint8\_t \*response)  
*Generates the response based on Key and Challenge provided.*
- [ATCA\\_STATUS sha206a\\_authenticate](#) (uint8\_t \*challenge, uint8\_t \*expected\_response, uint8\_t \*is\_authenticated)  
*verifies the challenge and provided response using key in device*
- [ATCA\\_STATUS sha206a\\_verify\\_device\\_consumption](#) (uint8\_t \*is\_consumed)  
*verifies the device is fully consumed or not based on Parent and Derived Key use flags.*
- [ATCA\\_STATUS sha206a\\_check\\_dk\\_useflag\\_validity](#) (uint8\_t \*is\_consumed)  
*verifies Derived Key use flags for consumption*
- [ATCA\\_STATUS sha206a\\_check\\_pk\\_useflag\\_validity](#) (uint8\_t \*is\_consumed)  
*verifies Parent Key use flags for consumption*
- [ATCA\\_STATUS sha206a\\_get\\_dk\\_useflag\\_count](#) (uint8\_t \*dk\_available\_count)  
*calculates available Derived Key use counts*
- [ATCA\\_STATUS sha206a\\_get\\_pk\\_useflag\\_count](#) (uint8\_t \*pk\_available\_count)  
*calculates available Parent Key use counts*
- [ATCA\\_STATUS sha206a\\_get\\_dk\\_update\\_count](#) (uint8\_t \*dk\_update\_count)  
*Read Derived Key slot update count. It will be wraps around 256.*

- [ATCA\\_STATUS sha206a\\_write\\_data\\_store](#) (uint8\_t slot, uint8\_t \*data, uint8\_t block, uint8\_t offset, uint8\_t len, bool lock\_after\_write)  
*Update the data store slot with user data and lock it if necessary.*
- [ATCA\\_STATUS sha206a\\_read\\_data\\_store](#) (uint8\_t slot, uint8\_t \*data, uint8\_t offset, uint8\_t len)  
*Read the data stored in Data store.*
- [ATCA\\_STATUS sha206a\\_get\\_data\\_store\\_lock\\_status](#) (uint8\_t slot, uint8\_t \*is\_locked)  
*Returns the lock status of the given data store.*

### 20.1.1 Detailed Description

Provides APIs to use with ATSHA206A device.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.1.2 Function Documentation

#### 20.1.2.1 sha206a\_authenticate()

```
ATCA_STATUS sha206a_authenticate (
    uint8_t * challenge,
    uint8_t * expected_response,
    uint8_t * is_authenticated )
```

verifies the challenge and provided response using key in device

#### Parameters

in	<i>challenge</i>	Challenge to be used in the response calculations
in	<i>expected_response</i>	Expected response from the device.
out	<i>is_authenticated</i>	result of expected of response and calcaulted response

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.1.2.2 sha206a\_check\_dk\_useflag\_validity()

```
ATCA_STATUS sha206a_check_dk_useflag_validity (
    uint8_t * is_consumed )
```

verifies Derived Key use flags for consumption

### Parameters

out	<i>is_consumed</i>	indicates if DK is available for consumption.
-----	--------------------	---

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.1.2.3 sha206a\_check\_pk\_useflag\_validity()

```
ATCA_STATUS sha206a_check_pk_useflag_validity (
    uint8_t * is_consumed )
```

verifies Parent Key use flags for consumption

### Parameters

out	<i>is_consumed</i>	indicates if PK is available for consumption
-----	--------------------	--

### Returns

ATCA\_SUCCESS on success, otherwise an error code

#### 20.1.2.4 sha206a\_diversify\_parent\_key()

```
ATCA_STATUS sha206a_diversify_parent_key (
    uint8_t * parent_key,
    uint8_t * diversified_key )
```

Computes the diversified key based on the parent key provided and device serial number.

### Parameters

in	<i>parent_key</i>	parent key to be diversified
out	<i>diversified_key</i>	diversified parent key

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.1.2.5 sha206a\_generate\_challenge\_response\_pair()

```
ATCA_STATUS sha206a_generate_challenge_response_pair (
    uint8_t * key,
    uint8_t * challenge,
    uint8_t * response )
```

Generates the response based on Key and Challenge provided.

#### Parameters

in	<i>key</i>	Input data contains device's key
in	<i>challenge</i>	Input data to be used in challenge response calculation
out	<i>response</i>	response derived from key and challenge

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.1.2.6 sha206a\_generate\_derive\_key()

```
ATCA_STATUS sha206a_generate_derive_key (
    uint8_t * parent_key,
    uint8_t * derived_key,
    uint8_t param1,
    uint16_t param2 )
```

Generates the derived key based on the parent key and other parameters provided.

#### Parameters

in	<i>parent_key</i>	Input data contains device's parent key
out	<i>derived_key</i>	Output data derived from parent key
in	<i>param1</i>	Input data to be used in derive key calculation
in	<i>param2</i>	Input data to be used in derive key calculation

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.1.2.7 sha206a\_get\_data\_store\_lock\_status()

```
ATCA_STATUS sha206a_get_data_store_lock_status (
    uint8_t slot,
    uint8_t * is_locked )
```

Returns the lock status of the given data store.

### Parameters

in	<i>slot</i>	Slot number of the data store
out	<i>is_locked</i>	lock status of the data store

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.1.2.8 sha206a\_get\_dk\_update\_count()

```
ATCA_STATUS sha206a_get_dk_update_count (
    uint8_t * dk_update_count )
```

Read Derived Key slot update count. It will be wraps around 256.

### Parameters

out	<i>dk_update_count</i>	returns number of times the slot has been updated with derived key
-----	------------------------	--

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.1.2.9 sha206a\_get\_dk\_useflag\_count()

```
ATCA_STATUS sha206a_get_dk_useflag_count (
    uint8_t * dk_available_count )
```

calculates available Derived Key use counts

### Parameters

out	<i>dk_available_count</i>	counts available bit's as 1
-----	---------------------------	-----------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.1.2.10 sha206a\_get\_pk\_useflag\_count()

```
ATCA_STATUS sha206a_get_pk_useflag_count (
    uint8_t * pk_available_count )
```

calculates available Parent Key use counts

#### Parameters

out	<i>pk_available_count</i>	counts available bit's as 1
-----	---------------------------	-----------------------------

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.1.2.11 sha206a\_read\_data\_store()

```
ATCA_STATUS sha206a_read_data_store (
    uint8_t slot,
    uint8_t * data,
    uint8_t offset,
    uint8_t len )
```

Read the data stored in Data store.

#### Parameters

in	<i>slot</i>	Slot number to read from
in	<i>data</i>	Pointer to hold slot data data
in	<i>offset</i>	Byte offset within the zone to read from.
in	<i>len</i>	data length

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.1.2.12 sha206a\_verify\_device\_consumption()

```
ATCA_STATUS sha206a_verify_device_consumption (
    uint8_t * is_consumed )
```

verifies the device is fully consumed or not based on Parent and Derived Key use flags.

#### Parameters

out	<i>is_consumed</i>	result of device consumption
-----	--------------------	------------------------------

## 20.2 api\_206a.h File Reference

---

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.1.2.13 sha206a\_write\_data\_store()

```
ATCA_STATUS sha206a_write_data_store (
    uint8_t slot,
    uint8_t * data,
    uint8_t block,
    uint8_t offset,
    uint8_t len,
    bool lock_after_write )
```

Update the data store slot with user data and lock it if necessary.

### Parameters

in	<i>slot</i>	Slot number to be written with data
in	<i>data</i>	Pointer that holds the data
in	<i>block</i>	32-byte block to write to.
in	<i>offset</i>	4-byte word within the specified block to write to. If performing a 32-byte write, this should be 0.
in	<i>len</i>	data length
in	<i>lock_after_write</i>	set 1 to lock slot after write, otherwise 0

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.2 api\_206a.h File Reference

Provides api interfaces to use with ATSHA206A device.

```
#include "atca_status.h"
#include "atca_command.h"
```

### Macros

- #define ATCA\_SHA206A\_ZONE\_WRITE\_LOCK 0x20
- #define ATCA\_SHA206A\_DKEY\_CONSUMPTION\_MASK 0x01
- #define ATCA\_SHA206A\_PKEY\_CONSUMPTION\_MASK 0x02
- #define ATCA\_SHA206A\_SYMMETRIC\_KEY\_ID\_SLOT 0x07

### Enumerations

- enum { SHA206A\_DATA\_STORE0 =8, SHA206A\_DATA\_STORE1, SHA206A\_DATA\_STORE2 }



## Functions

- [ATCA\\_STATUS sha206a\\_diversify\\_parent\\_key](#) (uint8\_t \*parent\_key, uint8\_t \*diversified\_key)  
*Computes the diversified key based on the parent key provided and device serial number.*
- [ATCA\\_STATUS sha206a\\_generate\\_derive\\_key](#) (uint8\_t \*parent\_key, uint8\_t \*derived\_key, uint8\_t param1, uint16\_t param2)  
*Generates the derived key based on the parent key and other parameters provided.*
- [ATCA\\_STATUS sha206a\\_generate\\_challenge\\_response\\_pair](#) (uint8\_t \*key, uint8\_t \*challenge, uint8\_t \*response)  
*Generates the response based on Key and Challenge provided.*
- [ATCA\\_STATUS sha206a\\_authenticate](#) (uint8\_t \*challenge, uint8\_t \*expected\_response, uint8\_t \*is\_authenticated)  
*verifies the challenge and provided response using key in device*
- [ATCA\\_STATUS sha206a\\_verify\\_device\\_consumption](#) (uint8\_t \*is\_consumed)  
*verifies the device is fully consumed or not based on Parent and Derived Key use flags.*
- [ATCA\\_STATUS sha206a\\_check\\_dk\\_useflag\\_validity](#) (uint8\_t \*is\_valid)  
*verifies Derived Key use flags for consumption*
- [ATCA\\_STATUS sha206a\\_check\\_pk\\_useflag\\_validity](#) (uint8\_t \*is\_valid)  
*verifies Parent Key use flags for consumption*
- [ATCA\\_STATUS sha206a\\_get\\_dk\\_useflag\\_count](#) (uint8\_t \*dk\_available\_count)  
*calculates available Derived Key use counts*
- [ATCA\\_STATUS sha206a\\_get\\_pk\\_useflag\\_count](#) (uint8\_t \*pk\_available\_count)  
*calculates available Parent Key use counts*
- [ATCA\\_STATUS sha206a\\_get\\_dk\\_update\\_count](#) (uint8\_t \*dk\_update\_count)  
*Read Derived Key slot update count. It will be wraps around 256.*
- [ATCA\\_STATUS sha206a\\_write\\_data\\_store](#) (uint8\_t slot, uint8\_t \*data, uint8\_t block, uint8\_t offset, uint8\_t len, bool lock\_after\_write)  
*Update the data store slot with user data and lock it if necessary.*
- [ATCA\\_STATUS sha206a\\_read\\_data\\_store](#) (uint8\_t slot, uint8\_t \*data, uint8\_t offset, uint8\_t len)  
*Read the data stored in Data store.*
- [ATCA\\_STATUS sha206a\\_get\\_data\\_store\\_lock\\_status](#) (uint8\_t slot, uint8\_t \*is\_locked)  
*Returns the lock status of the given data store.*

### 20.2.1 Detailed Description

Provides api interfaces to use with ATSHA206A device.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.2.2 Macro Definition Documentation

#### 20.2.2.1 ATCA\_SHA206A\_DKEY\_CONSUMPTION\_MASK

```
#define ATCA_SHA206A_DKEY_CONSUMPTION_MASK 0x01
```

### 20.2.2.2 ATCA\_SHA206A\_PKEY\_CONSUMPTION\_MASK

```
#define ATCA_SHA206A_PKEY_CONSUMPTION_MASK 0x02
```

### 20.2.2.3 ATCA\_SHA206A\_SYMMETRIC\_KEY\_ID\_SLOT

```
#define ATCA_SHA206A_SYMMETRIC_KEY_ID_SLOT 0x07
```

### 20.2.2.4 ATCA\_SHA206A\_ZONE\_WRITE\_LOCK

```
#define ATCA_SHA206A_ZONE_WRITE_LOCK 0x20
```

## 20.2.3 Enumeration Type Documentation

### 20.2.3.1 anonymous enum

anonymous enum

#### Enumerator

SHA206A_DATA_STORE0	
SHA206A_DATA_STORE1	
SHA206A_DATA_STORE2	

## 20.2.4 Function Documentation

### 20.2.4.1 sha206a\_authenticate()

```
ATCA_STATUS sha206a_authenticate (
    uint8_t * challenge,
    uint8_t * expected_response,
    uint8_t * is_authenticated )
```

verifies the challenge and provided response using key in device

**Parameters**

in	<i>challenge</i>	Challenge to be used in the response calculations
in	<i>expected_response</i>	Expected response from the device.
out	<i>is_authenticated</i>	result of expected of response and calcaulted response

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.2.4.2 sha206a\_check\_dk\_useflag\_validity()**

```
ATCA_STATUS sha206a_check_dk_useflag_validity (
    uint8_t * is_consumed )
```

verifies Derived Key use flags for consumption

**Parameters**

out	<i>is_consumed</i>	indicates if DK is available for consumption.
-----	--------------------	---

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.2.4.3 sha206a\_check\_pk\_useflag\_validity()**

```
ATCA_STATUS sha206a_check_pk_useflag_validity (
    uint8_t * is_consumed )
```

verifies Parent Key use flags for consumption

**Parameters**

out	<i>is_consumed</i>	indicates if PK is available for consumption
-----	--------------------	--

**Returns**

ATCA\_SUCCESS on success, otherwise an error code

#### 20.2.4.4 sha206a\_diversify\_parent\_key()

```
ATCA_STATUS sha206a_diversify_parent_key (
    uint8_t * parent_key,
    uint8_t * diversified_key )
```

Computes the diversified key based on the parent key provided and device serial number.

##### Parameters

in	<i>parent_key</i>	parent key to be diversified
out	<i>diversified_key</i>	diversified parent key

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.2.4.5 sha206a\_generate\_challenge\_response\_pair()

```
ATCA_STATUS sha206a_generate_challenge_response_pair (
    uint8_t * key,
    uint8_t * challenge,
    uint8_t * response )
```

Generates the response based on Key and Challenge provided.

##### Parameters

in	<i>key</i>	Input data contains device's key
in	<i>challenge</i>	Input data to be used in challenge response calculation
out	<i>response</i>	response derived from key and challenge

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.2.4.6 sha206a\_generate\_derive\_key()

```
ATCA_STATUS sha206a_generate_derive_key (
    uint8_t * parent_key,
    uint8_t * derived_key,
    uint8_t param1,
    uint16_t param2 )
```

Generates the derived key based on the parent key and other parameters provided.

**Parameters**

in	<i>parent_key</i>	Input data contains device's parent key
out	<i>derived_key</i>	Output data derived from parent key
in	<i>param1</i>	Input data to be used in derive key calculation
in	<i>param2</i>	Input data to be used in derive key calculation

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.2.4.7 sha206a\_get\_data\_store\_lock\_status()**

```
ATCA_STATUS sha206a_get_data_store_lock_status (
    uint8_t slot,
    uint8_t * is_locked )
```

Returns the lock status of the given data store.

**Parameters**

in	<i>slot</i>	Slot number of the data store
out	<i>is_locked</i>	lock status of the data store

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.2.4.8 sha206a\_get\_dk\_update\_count()**

```
ATCA_STATUS sha206a_get_dk_update_count (
    uint8_t * dk_update_count )
```

Read Derived Key slot update count. It will be wraps around 256.

**Parameters**

out	<i>dk_update_count</i>	returns number of times the slot has been updated with derived key
-----	------------------------	--

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

### 20.2.4.9 sha206a\_get\_dk\_useflag\_count()

```
ATCA_STATUS sha206a_get_dk_useflag_count (
    uint8_t * dk_available_count )
```

calculates available Derived Key use counts

#### Parameters

out	<i>dk_available_count</i>	counts available bit's as 1
-----	---------------------------	-----------------------------

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.2.4.10 sha206a\_get\_pk\_useflag\_count()

```
ATCA_STATUS sha206a_get_pk_useflag_count (
    uint8_t * pk_available_count )
```

calculates available Parent Key use counts

#### Parameters

out	<i>pk_available_count</i>	counts available bit's as 1
-----	---------------------------	-----------------------------

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.2.4.11 sha206a\_read\_data\_store()

```
ATCA_STATUS sha206a_read_data_store (
    uint8_t slot,
    uint8_t * data,
    uint8_t offset,
    uint8_t len )
```

Read the data stored in Data store.

#### Parameters

in	<i>slot</i>	Slot number to read from
in	<i>data</i>	Pointer to hold slot data data
in	<i>offset</i>	Byte offset within the zone to read from.
in	<i>len</i>	data length

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.2.4.12 sha206a\_verify\_device\_consumption()**

```
ATCA_STATUS sha206a_verify_device_consumption (
    uint8_t * is_consumed )
```

verifies the device is fully consumed or not based on Parent and Derived Key use flags.

**Parameters**

out	<i>is_consumed</i>	result of device consumption
-----	--------------------	------------------------------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.2.4.13 sha206a\_write\_data\_store()**

```
ATCA_STATUS sha206a_write_data_store (
    uint8_t slot,
    uint8_t * data,
    uint8_t block,
    uint8_t offset,
    uint8_t len,
    bool lock_after_write )
```

Update the data store slot with user data and lock it if necessary.

**Parameters**

in	<i>slot</i>	Slot number to be written with data
in	<i>data</i>	Pointer that holds the data
in	<i>block</i>	32-byte block to write to.
in	<i>offset</i>	4-byte word within the specified block to write to. If performing a 32-byte write, this should be 0.
in	<i>len</i>	data length
in	<i>lock_after_write</i>	set 1 to lock slot after write, otherwise 0

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

## 20.3 atca\_basic.c File Reference

CryptoAuthLib Basic API methods. These methods provide a simpler way to access the core crypto methods.

```
#include "atca_basic.h"
#include "atca_version.h"
```

### Functions

- [ATCA\\_STATUS atcab\\_version](#) (char \*ver\_str)  
*basic API methods are all prefixed with atcab\_ (CryptoAuthLib Basic) the fundamental premise of the basic API is it is based on a single interface instance and that instance is global, so all basic API commands assume that one global device is the one to operate on.*
- [ATCA\\_STATUS atcab\\_init\\_ext](#) (ATCADevice \*device, ATCAIfaceCfg \*cfg)  
*Creates and initializes a ATCADevice context.*
- [ATCA\\_STATUS atcab\\_init](#) (ATCAIfaceCfg \*cfg)  
*Creates a global ATCADevice object used by Basic API.*
- [ATCA\\_STATUS atcab\\_init\\_device](#) (ATCADevice ca\_device)  
*Initialize the global ATCADevice object to point to one of your choosing for use with all the atcab\_ basic API.*
- [ATCA\\_STATUS atcab\\_release\\_ext](#) (ATCADevice \*device)  
*release (free) the an ATCADevice instance.*
- [ATCA\\_STATUS atcab\\_release](#) (void)  
*release (free) the global ATCADevice instance. This must be called in order to release or free up the interface.*
- [ATCADevice atcab\\_get\\_device](#) (void)  
*Get the global device object.*
- [ATCADeviceType atcab\\_get\\_device\\_type\\_ext](#) (ATCADevice device)  
*Get the selected device type of rthe device context.*
- [ATCADeviceType atcab\\_get\\_device\\_type](#) (void)  
*Get the current device type configured for the global ATCADevice.*
- [bool atcab\\_is\\_ca\\_device](#) (ATCADeviceType dev\_type)  
*Check whether the device is cryptoauth device.*
- [bool atcab\\_is\\_ta\\_device](#) (ATCADeviceType dev\_type)  
*Check whether the device is Trust Anchor device.*
- [ATCA\\_STATUS atcab\\_wakeup](#) (void)  
*wakeup the CryptoAuth device*
- [ATCA\\_STATUS atcab\\_idle](#) (void)  
*idle the CryptoAuth device*
- [ATCA\\_STATUS atcab\\_sleep](#) (void)  
*invoke sleep on the CryptoAuth device*
- [ATCA\\_STATUS atcab\\_get\\_zone\\_size](#) (uint8\_t zone, uint16\_t slot, size\_t \*size)  
*Gets the size of the specified zone in bytes.*
- [ATCA\\_STATUS atcab\\_aes](#) (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*aes\_in, uint8\_t \*aes\_out)  
*Compute the AES-128 encrypt, decrypt, or GFM calculation.*
- [ATCA\\_STATUS atcab\\_aes\\_encrypt\\_ext](#) (ATCADevice device, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*plaintext, uint8\_t \*ciphertext)  
*Perform an AES-128 encrypt operation with a key in the device.*
- [ATCA\\_STATUS atcab\\_aes\\_encrypt](#) (uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*plaintext, uint8\_t \*ciphertext)  
*Perform an AES-128 encrypt operation with a key in the device.*



- [ATCA\\_STATUS atcab\\_aes\\_decrypt\\_ext](#) ([ATCADevice](#) device, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*ciphertext, uint8\_t \*plaintext)  
*Perform an AES-128 decrypt operation with a key in the device.*
- [ATCA\\_STATUS atcab\\_aes\\_decrypt](#) (uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*ciphertext, uint8\_t \*plaintext)  
*Perform an AES-128 decrypt operation with a key in the device.*
- [ATCA\\_STATUS atcab\\_aes\\_gfm](#) (const uint8\_t \*h, const uint8\_t \*input, uint8\_t \*output)  
*Perform a Galois Field Multiply (GFM) operation.*
- [ATCA\\_STATUS atcab\\_aes\\_gcm\\_init](#) ([atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*iv, size\_t iv\_size)  
*Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.*
- [ATCA\\_STATUS atcab\\_aes\\_gcm\\_init\\_rand](#) ([atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, uint16\_t key\_id, uint8\_t key\_block, size\_t rand\_size, const uint8\_t \*free\_field, size\_t free\_field\_size, uint8\_t \*iv)  
*Initialize context for AES GCM operation with a IV composed of a random and optional fixed(free) field, which is common when starting an encrypt operation.*
- [ATCA\\_STATUS atcab\\_aes\\_gcm\\_aad\\_update](#) ([atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*aad, uint32\_t aad\_size)  
*Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.*
- [ATCA\\_STATUS atcab\\_aes\\_gcm\\_encrypt\\_update](#) ([atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*plaintext, uint32\_t plaintext\_size, uint8\_t \*ciphertext)  
*Encrypt data using GCM mode and a key within the ATECC608 device. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function.*
- [ATCA\\_STATUS atcab\\_aes\\_gcm\\_encrypt\\_finish](#) ([atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, uint8\_t \*tag, size\_t tag\_size)  
*Complete a GCM encrypt operation returning the authentication tag.*
- [ATCA\\_STATUS atcab\\_aes\\_gcm\\_decrypt\\_update](#) ([atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*ciphertext, uint32\_t ciphertext\_size, uint8\_t \*plaintext)  
*Decrypt data using GCM mode and a key within the ATECC608 device. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function.*
- [ATCA\\_STATUS atcab\\_aes\\_gcm\\_decrypt\\_finish](#) ([atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*tag, size\_t tag\_size, bool \*is\_verified)  
*Complete a GCM decrypt operation verifying the authentication tag.*
- [ATCA\\_STATUS atcab\\_checkmac](#) (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*challenge, const uint8\_t \*response, const uint8\_t \*other\_data)  
*Compares a MAC response with input values.*
- [ATCA\\_STATUS atcab\\_counter](#) (uint8\_t mode, uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Compute the Counter functions.*
- [ATCA\\_STATUS atcab\\_counter\\_increment](#) (uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Increments one of the device's monotonic counters.*
- [ATCA\\_STATUS atcab\\_counter\\_read](#) (uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Read one of the device's monotonic counters.*
- [ATCA\\_STATUS atcab\\_derivekey](#) (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*mac)  
*Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.*
- [ATCA\\_STATUS atcab\\_ecdh\\_base](#) (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, uint8\_t \*out\_nonce)  
*Base function for generating premaster secret key using ECDH.*
- [ATCA\\_STATUS atcab\\_ecdh](#) (uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms)  
*ECDH command with a private key in a slot and the premaster secret is returned in the clear.*
- [ATCA\\_STATUS atcab\\_ecdh\\_enc](#) (uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*read\_key, uint16\_t read\_key\_id, const uint8\_t num\_in[(20)])  
*ECDH command with a private key in a slot and the premaster secret is read from the next slot.*
- [ATCA\\_STATUS atcab\\_ecdh\\_ioenc](#) (uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*io\_key)

- ECDH command with a private key in a slot and the premaster secret is returned encrypted using the IO protection key.*
- **ATCA\_STATUS atcab\_ecdh\_tempkey** (const uint8\_t \*public\_key, uint8\_t \*pms)  
*ECDH command with a private key in TempKey and the premaster secret is returned in the clear.*
  - **ATCA\_STATUS atcab\_ecdh\_tempkey\_ioenc** (const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*io\_key)  
*ECDH command with a private key in TempKey and the premaster secret is returned encrypted using the IO protection key.*
  - **ATCA\_STATUS atcab\_gendig** (uint8\_t zone, uint16\_t key\_id, const uint8\_t \*other\_data, uint8\_t other\_data\_size)  
*Issues a GenDig command, which performs a SHA256 hash on the source data indicated by zone with the contents of TempKey. See the CryptoAuth datasheet for your chip to see what the values of zone correspond to.*
  - **ATCA\_STATUS atcab\_genkey\_base** (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*other\_data, uint8\_t \*public\_key)  
*Issues GenKey command, which can generate a private key, compute a public key, and/or compute a digest of a public key.*
  - **ATCA\_STATUS atcab\_genkey** (uint16\_t key\_id, uint8\_t \*public\_key)  
*Issues GenKey command, which generates a new random private key in slot/handle and returns the public key.*
  - **ATCA\_STATUS atcab\_get\_pubkey** (uint16\_t key\_id, uint8\_t \*public\_key)  
*Uses GenKey command to calculate the public key from an existing private key in a slot.*
  - **ATCA\_STATUS atcab\_hmac** (uint8\_t mode, uint16\_t key\_id, uint8\_t \*digest)  
*Issues a HMAC command, which computes an HMAC/SHA-256 digest of a key stored in the device, a challenge, and other information on the device.*
  - **ATCA\_STATUS atcab\_info\_base** (uint8\_t mode, uint16\_t param2, uint8\_t \*out\_data)  
*Issues an Info command, which return internal device information and can control GPIO and the persistent latch.*
  - **ATCA\_STATUS atcab\_info** (uint8\_t \*revision)  
*Use the Info command to get the device revision (DevRev).*
  - **ATCA\_STATUS atcab\_info\_set\_latch** (bool state)  
*Use the Info command to set the persistent latch state for an ATECC608 device.*
  - **ATCA\_STATUS atcab\_info\_get\_latch** (bool \*state)  
*Use the Info command to get the persistent latch current state for an ATECC608 device.*
  - **ATCA\_STATUS atcab\_kdf** (uint8\_t mode, uint16\_t key\_id, const uint32\_t details, const uint8\_t \*message, uint8\_t \*out\_data, uint8\_t \*out\_nonce)  
*Executes the KDF command, which derives a new key in PRF, AES, or HKDF modes.*
  - **ATCA\_STATUS atcab\_lock** (uint8\_t mode, uint16\_t summary\_crc)  
*The Lock command prevents future modifications of the Configuration and/or Data and OTP zones. If the device is so configured, then this command can be used to lock individual data slots. This command fails if the designated area is already locked.*
  - **ATCA\_STATUS atcab\_lock\_config\_zone** (void)  
*Unconditionally (no CRC required) lock the config zone.*
  - **ATCA\_STATUS atcab\_lock\_config\_zone\_crc** (uint16\_t summary\_crc)  
*Lock the config zone with summary CRC.*
  - **ATCA\_STATUS atcab\_lock\_data\_zone** (void)  
*Unconditionally (no CRC required) lock the data zone (slots and OTP). for CryptoAuth devices and lock the setup for Trust Anchor device.*
  - **ATCA\_STATUS atcab\_lock\_data\_zone\_crc** (uint16\_t summary\_crc)  
*Lock the data zone (slots and OTP) with summary CRC.*
  - **ATCA\_STATUS atcab\_lock\_data\_slot** (uint16\_t slot)  
*Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (KeyConfig.Lockable=1) (for cryptoauth devices) or Lock an individual handle in shared data element on an Trust Anchor device (for Trust Anchor devices).*
  - **ATCA\_STATUS atcab\_mac** (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*challenge, uint8\_t \*digest)  
*Executes MAC command, which computes a SHA-256 digest of a key stored in the device, a challenge, and other information on the device.*

- **ATCA\_STATUS atcab\_nonce\_base** (uint8\_t mode, uint16\_t zero, const uint8\_t \*num\_in, uint8\_t \*rand\_out)  
*Executes Nonce command, which loads a random or fixed nonce/data into the device for use by subsequent commands.*
- **ATCA\_STATUS atcab\_nonce** (const uint8\_t \*num\_in)  
*Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.*
- **ATCA\_STATUS atcab\_nonce\_load** (uint8\_t target, const uint8\_t \*num\_in, uint16\_t num\_in\_size)  
*Execute a Nonce command in pass-through mode to load one of the device's internal buffers with a fixed value.*
- **ATCA\_STATUS atcab\_nonce\_rand** (const uint8\_t \*num\_in, uint8\_t \*rand\_out)  
*Execute a Nonce command to generate a random nonce combining a host nonce (num\_in) and a device random number.*
- **ATCA\_STATUS atcab\_challenge** (const uint8\_t \*num\_in)  
*Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.*
- **ATCA\_STATUS atcab\_challenge\_seed\_update** (const uint8\_t \*num\_in, uint8\_t \*rand\_out)  
*Execute a Nonce command to generate a random challenge combining a host nonce (num\_in) and a device random number.*
- **ATCA\_STATUS atcab\_priv\_write** (uint16\_t key\_id, const uint8\_t priv\_key[36], uint16\_t write\_key\_id, const uint8\_t write\_key[32], const uint8\_t num\_in[(20)])  
*Executes PrivWrite command, to write externally generated ECC private keys into the device.*
- **ATCA\_STATUS atcab\_random\_ext** (ATCADevice device, uint8\_t \*rand\_out)  
*Executes Random command, which generates a 32 byte random number from the device.*
- **ATCA\_STATUS atcab\_random** (uint8\_t \*rand\_out)  
*Executes Random command, which generates a 32 byte random number from the device.*
- **ATCA\_STATUS atcab\_read\_zone** (uint8\_t zone, uint16\_t slot, uint8\_t block, uint8\_t offset, uint8\_t \*data, uint8\_t len)  
*Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.*
- **ATCA\_STATUS atcab\_is\_locked** (uint8\_t zone, bool \*is\_locked)  
*Executes Read command, which reads the configuration zone to see if the specified zone is locked.*
- **ATCA\_STATUS atcab\_is\_config\_locked** (bool \*is\_locked)  
*This function check whether configuration zone is locked or not.*
- **ATCA\_STATUS atcab\_is\_data\_locked** (bool \*is\_locked)  
*This function check whether data/setup zone is locked or not.*
- **ATCA\_STATUS atcab\_is\_slot\_locked** (uint16\_t slot, bool \*is\_locked)  
*This function check whether slot/handle is locked or not.*
- **ATCA\_STATUS atcab\_read\_bytes\_zone** (uint8\_t zone, uint16\_t slot, size\_t offset, uint8\_t \*data, size\_t length)  
*Used to read an arbitrary number of bytes from any zone configured for clear reads.*
- **ATCA\_STATUS atcab\_read\_serial\_number** (uint8\_t \*serial\_number)  
*This function returns serial number of the device.*
- **ATCA\_STATUS atcab\_read\_pubkey** (uint16\_t slot, uint8\_t \*public\_key)  
*Executes Read command to read an ECC P256 public key from a slot configured for clear reads.*
- **ATCA\_STATUS atcab\_read\_sig** (uint16\_t slot, uint8\_t \*sig)  
*Executes Read command to read a 64 byte ECDSA P256 signature from a slot configured for clear reads.*
- **ATCA\_STATUS atcab\_read\_config\_zone** (uint8\_t \*config\_data)  
*Executes Read command to read the complete device configuration zone.*
- **ATCA\_STATUS atcab\_cmp\_config\_zone** (uint8\_t \*config\_data, bool \*same\_config)  
*Compares a specified configuration zone with the configuration zone currently on the device.*
- **ATCA\_STATUS atcab\_read\_enc** (uint16\_t key\_id, uint8\_t block, uint8\_t \*data, const uint8\_t \*enc\_key, const uint16\_t enc\_key\_id, const uint8\_t num\_in[(20)])  
*Executes Read command on a slot configured for encrypted reads and decrypts the data to return it as plaintext.*
- **ATCA\_STATUS atcab\_secureboot** (uint8\_t mode, uint16\_t param2, const uint8\_t \*digest, const uint8\_t \*signature, uint8\_t \*mac)

- Executes Secure Boot command, which provides support for secure boot of an external MCU or MPU.*
- **ATCA\_STATUS atcab\_secureboot\_mac** (uint8\_t mode, const uint8\_t \*digest, const uint8\_t \*signature, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)  
*Executes Secure Boot command with encrypted digest and validated MAC response using the IO protection key.*
  - **ATCA\_STATUS atcab\_selftest** (uint8\_t mode, uint16\_t param2, uint8\_t \*result)  
*Executes the SelfTest command, which performs a test of one or more of the cryptographic engines within the ATCC608 chip.*
  - **ATCA\_STATUS atcab\_sha\_base** (uint8\_t mode, uint16\_t length, const uint8\_t \*data\_in, uint8\_t \*data\_out, uint16\_t \*data\_out\_size)  
*Executes SHA command, which computes a SHA-256 or HMAC/SHA-256 digest for general purpose use by the host system.*
  - **ATCA\_STATUS atcab\_sha\_start** (void)  
*Executes SHA command to initialize SHA-256 calculation engine.*
  - **ATCA\_STATUS atcab\_sha\_update** (const uint8\_t \*message)  
*Executes SHA command to add 64 bytes of message data to the current context.*
  - **ATCA\_STATUS atcab\_sha\_end** (uint8\_t \*digest, uint16\_t length, const uint8\_t \*message)  
*Executes SHA command to complete SHA-256 or HMAC/SHA-256 operation.*
  - **ATCA\_STATUS atcab\_sha\_read\_context** (uint8\_t \*context, uint16\_t \*context\_size)  
*Executes SHA command to read the SHA-256 context back. Only for ATECC608 with SHA-256 contexts. HMAC not supported.*
  - **ATCA\_STATUS atcab\_sha\_write\_context** (const uint8\_t \*context, uint16\_t context\_size)  
*Executes SHA command to write (restore) a SHA-256 context into the device. Only supported for ATECC608 with SHA-256 contexts.*
  - **ATCA\_STATUS atcab\_sha** (uint16\_t length, const uint8\_t \*message, uint8\_t \*digest)  
*Use the SHA command to compute a SHA-256 digest.*
  - **ATCA\_STATUS atcab\_hw\_sha2\_256** (const uint8\_t \*data, size\_t data\_size, uint8\_t \*digest)  
*Use the SHA command to compute a SHA-256 digest.*
  - **ATCA\_STATUS atcab\_hw\_sha2\_256\_init** (atca\_sha256\_ctx\_t \*ctx)  
*Initialize a SHA context for performing a hardware SHA-256 operation on a device. Note that only one SHA operation can be run at a time.*
  - **ATCA\_STATUS atcab\_hw\_sha2\_256\_update** (atca\_sha256\_ctx\_t \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Add message data to a SHA context for performing a hardware SHA-256 operation on a device.*
  - **ATCA\_STATUS atcab\_hw\_sha2\_256\_finish** (atca\_sha256\_ctx\_t \*ctx, uint8\_t \*digest)  
*Finish SHA-256 digest for a SHA context for performing a hardware SHA-256 operation on a device.*
  - **ATCA\_STATUS atcab\_sha\_hmac\_init** (atca\_hmac\_sha256\_ctx\_t \*ctx, uint16\_t key\_slot)  
*Executes SHA command to start an HMAC/SHA-256 operation.*
  - **ATCA\_STATUS atcab\_sha\_hmac\_update** (atca\_hmac\_sha256\_ctx\_t \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Executes SHA command to add an arbitrary amount of message data to a HMAC/SHA-256 operation.*
  - **ATCA\_STATUS atcab\_sha\_hmac\_finish** (atca\_hmac\_sha256\_ctx\_t \*ctx, uint8\_t \*digest, uint8\_t target)  
*Executes SHA command to complete a HMAC/SHA-256 operation.*
  - **ATCA\_STATUS atcab\_sha\_hmac** (const uint8\_t \*data, size\_t data\_size, uint16\_t key\_slot, uint8\_t \*digest, uint8\_t target)  
*Use the SHA command to compute an HMAC/SHA-256 operation.*
  - **ATCA\_STATUS atcab\_sign\_base** (uint8\_t mode, uint16\_t key\_id, uint8\_t \*signature)  
*Executes the Sign command, which generates a signature using the ECDSA algorithm.*
  - **ATCA\_STATUS atcab\_sign** (uint16\_t key\_id, const uint8\_t \*msg, uint8\_t \*signature)  
*Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*
  - **ATCA\_STATUS atcab\_sign\_internal** (uint16\_t key\_id, bool is\_invalidate, bool is\_full\_sn, uint8\_t \*signature)  
*Executes Sign command to sign an internally generated message.*
  - **ATCA\_STATUS atcab\_updateextra** (uint8\_t mode, uint16\_t new\_value)

Executes UpdateExtra command to update the values of the two extra bytes within the Configuration zone (bytes 84 and 85).

- **ATCA\_STATUS atcab\_verify** (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*public\_key, const uint8\_t \*other\_data, uint8\_t \*mac)  
*Executes the Verify command, which takes an ECDSA [R,S] signature and verifies that it is correctly generated from a given message and public key. In all cases, the signature is an input to the command.*
- **ATCA\_STATUS atcab\_verify\_extern** (const uint8\_t \*message, const uint8\_t \*signature, const uint8\_t \*public\_key, bool \*is\_verified)  
*Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*
- **ATCA\_STATUS atcab\_verify\_extern\_mac** (const uint8\_t \*message, const uint8\_t \*signature, const uint8\_t \*public\_key, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)  
*Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. This function is only available on the ATECC608.*
- **ATCA\_STATUS atcab\_verify\_stored** (const uint8\_t \*message, const uint8\_t \*signature, uint16\_t key\_id, bool \*is\_verified)  
*Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*
- **ATCA\_STATUS atcab\_verify\_stored\_mac** (const uint8\_t \*message, const uint8\_t \*signature, uint16\_t key\_id, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)  
*Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with a public key stored in the device. This function is only available on the ATECC608.*
- **ATCA\_STATUS atcab\_verify\_validate** (uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*other\_data, bool \*is\_verified)  
*Executes the Verify command in Validate mode to validate a public key stored in a slot.*
- **ATCA\_STATUS atcab\_verify\_invalidate** (uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*other\_data, bool \*is\_verified)  
*Executes the Verify command in Invalidate mode which invalidates a previously validated public key stored in a slot.*
- **ATCA\_STATUS atcab\_write** (uint8\_t zone, uint16\_t address, const uint8\_t \*value, const uint8\_t \*mac)  
*Executes the Write command, which writes either one four byte word or a 32-byte block to one of the EEPROM zones on the device. Depending upon the value of the WriteConfig byte for this slot, the data may be required to be encrypted by the system prior to being sent to the device. This command cannot be used to write slots configured as ECC private keys.*
- **ATCA\_STATUS atcab\_write\_zone** (uint8\_t zone, uint16\_t slot, uint8\_t block, uint8\_t offset, const uint8\_t \*data, uint8\_t len)  
*Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.*
- **ATCA\_STATUS atcab\_write\_bytes\_zone** (uint8\_t zone, uint16\_t slot, size\_t offset\_bytes, const uint8\_t \*data, size\_t length)  
*Executes the Write command, which writes data into the configuration, otp, or data zones with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).*
- **ATCA\_STATUS atcab\_write\_pubkey** (uint16\_t slot, const uint8\_t \*public\_key)  
*Uses the write command to write a public key to a slot in the proper format.*
- **ATCA\_STATUS atcab\_write\_config\_zone** (const uint8\_t \*config\_data)  
*Executes the Write command, which writes the configuration zone.*
- **ATCA\_STATUS atcab\_write\_enc** (uint16\_t key\_id, uint8\_t block, const uint8\_t \*data, const uint8\_t \*enc\_key, const uint16\_t enc\_key\_id, const uint8\_t num\_in[(20)])  
*Executes the Write command, which performs an encrypted write of a 32 byte block into given slot.*
- **ATCA\_STATUS atcab\_write\_config\_counter** (uint16\_t counter\_id, uint32\_t counter\_value)  
*Initialize one of the monotonic counters in device with a specific value.*

## Variables

- const char **atca\_version** [] = "20200728"
- SHARED\_LIB\_EXPORT **ATCADevice\_gDevice** = NULL

### 20.3.1 Detailed Description

CryptoAuthLib Basic API methods. These methods provide a simpler way to access the core crypto methods.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.3.2 Variable Documentation

#### 20.3.2.1 \_gDevice

```
SHARED_LIB_EXPORT ATCADevice _gDevice = NULL
```

#### 20.3.2.2 atca\_version

```
const char atca_version[] = "20200728"
```

## 20.4 atca\_basic.h File Reference

CryptoAuthLib Basic API methods - a simple crypto authentication API. These methods manage a global ATCA↔Device object behind the scenes. They also manage the wake/idle state transitions so callers don't need to.

```
#include "cryptoauthlib.h"
#include "crypto/atca_crypto_sw_sha2.h"
#include "crypto/atca_crypto_hw_aes.h"
```

### Macros

- #define [atcab\\_cfg\\_discover\(...\)](#) [calib\\_cfg\\_discover\(\\_\\_VA\\_ARGS\\_\\_\)](#)
- #define [atcab\\_get\\_addr\(...\)](#) [calib\\_get\\_addr\(\\_\\_VA\\_ARGS\\_\\_\)](#)
- #define [atca\\_execute\\_command\(...\)](#) [calib\\_execute\\_command\(\\_\\_VA\\_ARGS\\_\\_\)](#)
- #define [SHA\\_CONTEXT\\_MAX\\_SIZE](#) (109)



## Functions

- **ATCA\_STATUS atcab\_version** (char \*ver\_str)  
*basic API methods are all prefixed with atcab\_ (CryptoAuthLib Basic) the fundamental premise of the basic API is it is based on a single interface instance and that instance is global, so all basic API commands assume that one global device is the one to operate on.*
- **ATCA\_STATUS atcab\_init\_ext** (ATCADevice \*device, ATCAfaceCfg \*cfg)  
*Creates and initializes a ATCADevice context.*
- **ATCA\_STATUS atcab\_init** (ATCAfaceCfg \*cfg)  
*Creates a global ATCADevice object used by Basic API.*
- **ATCA\_STATUS atcab\_init\_device** (ATCADevice ca\_device)  
*Initialize the global ATCADevice object to point to one of your choosing for use with all the atcab\_ basic API.*
- **ATCA\_STATUS atcab\_release\_ext** (ATCADevice \*device)  
*release (free) the an ATCADevice instance.*
- **ATCA\_STATUS atcab\_release** (void)  
*release (free) the global ATCADevice instance. This must be called in order to release or free up the interface.*
- **ATCADevice atcab\_get\_device** (void)  
*Get the global device object.*
- **ATCADeviceType atcab\_get\_device\_type\_ext** (ATCADevice device)  
*Get the selected device type of rthe device context.*
- **ATCADeviceType atcab\_get\_device\_type** (void)  
*Get the current device type configured for the global ATCADevice.*
- **bool atcab\_is\_ca\_device** (ATCADeviceType dev\_type)  
*Check whether the device is cryptoauth device.*
- **bool atcab\_is\_ta\_device** (ATCADeviceType dev\_type)  
*Check whether the device is Trust Anchor device.*
- **ATCA\_STATUS atcab\_aes\_cbc\_init\_ext** (ATCADevice device, atca\_aes\_cbc\_ctx\_t \*ctx, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*iv)  
*Initialize context for AES CBC operation.*
- **ATCA\_STATUS atcab\_aes\_cbc\_init** (atca\_aes\_cbc\_ctx\_t \*ctx, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*iv)  
*Initialize context for AES CBC operation.*
- **ATCA\_STATUS atcab\_aes\_cbc\_encrypt\_block\_ext** (atca\_aes\_cbc\_ctx\_t \*ctx, const uint8\_t \*plaintext, uint8\_t \*ciphertext)  
*Encrypt a block of data using CBC mode and a key within the device. atcab\_aes\_cbc\_init() should be called before the first use of this function.*
- **ATCA\_STATUS atcab\_aes\_cbc\_encrypt\_block** (atca\_aes\_cbc\_ctx\_t \*ctx, const uint8\_t \*plaintext, uint8\_t \*ciphertext)  
*Encrypt a block of data using CBC mode and a key within the device. atcab\_aes\_cbc\_init() should be called before the first use of this function.*
- **ATCA\_STATUS atcab\_aes\_cbc\_decrypt\_block** (atca\_aes\_cbc\_ctx\_t \*ctx, const uint8\_t \*ciphertext, uint8\_t \*plaintext)  
*Decrypt a block of data using CBC mode and a key within the device. atcab\_aes\_cbc\_init() should be called before the first use of this function.*
- **ATCA\_STATUS atcab\_aes\_cmac\_init\_ext** (ATCADevice device, atca\_aes\_cmac\_ctx\_t \*ctx, uint16\_t key\_id, uint8\_t key\_block)  
*Initialize a CMAC calculation using an AES-128 key in the device.*
- **ATCA\_STATUS atcab\_aes\_cmac\_init** (atca\_aes\_cmac\_ctx\_t \*ctx, uint16\_t key\_id, uint8\_t key\_block)  
*Initialize a CMAC calculation using an AES-128 key in the device.*
- **ATCA\_STATUS atcab\_aes\_cmac\_update** (atca\_aes\_cmac\_ctx\_t \*ctx, const uint8\_t \*data, uint32\_t data\_size)  
*Add data to an initialized CMAC calculation.*
- **ATCA\_STATUS atcab\_aes\_cmac\_finish** (atca\_aes\_cmac\_ctx\_t \*ctx, uint8\_t \*cmac, uint32\_t cmac\_size)  
*Finish a CMAC operation returning the CMAC value.*

- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_init\\_ext](#) (ATCADevice device, [atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, uint16\_t key\_id, uint8\_t key\_block, uint8\_t counter\_size, const uint8\_t \*iv)  
*Initialize context for AES CTR operation with an existing IV, which is common when start a decrypt operation.*
- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_init](#) (atca\_aes\_ctr\_ctx\_t \*ctx, uint16\_t key\_id, uint8\_t key\_block, uint8\_t counter\_size, const uint8\_t \*iv)  
*Initialize context for AES CTR operation with an existing IV, which is common when start a decrypt operation.*
- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_init\\_rand\\_ext](#) (ATCADevice device, [atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, uint16\_t key\_id, uint8\_t key\_block, uint8\_t counter\_size, uint8\_t \*iv)  
*Initialize context for AES CTR operation with a random nonce and counter set to 0 as the IV, which is common when starting an encrypt operation.*
- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_init\\_rand](#) (atca\_aes\_ctr\_ctx\_t \*ctx, uint16\_t key\_id, uint8\_t key\_block, uint8\_t counter\_size, uint8\_t \*iv)  
*Initialize context for AES CTR operation with a random nonce and counter set to 0 as the IV, which is common when starting an encrypt operation.*
- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_block](#) (atca\_aes\_ctr\_ctx\_t \*ctx, const uint8\_t \*input, uint8\_t \*output)  
*Process a block of data using CTR mode and a key within the device. [atcab\\_aes\\_ctr\\_init\(\)](#) or [atcab\\_aes\\_ctr\\_init\\_rand\(\)](#) should be called before the first use of this function.*
- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_encrypt\\_block](#) (atca\_aes\_ctr\_ctx\_t \*ctx, const uint8\_t \*plaintext, uint8\_t \*ciphertext)  
*Encrypt a block of data using CTR mode and a key within the device. [atcab\\_aes\\_ctr\\_init\(\)](#) or [atcab\\_aes\\_ctr\\_init\\_rand\(\)](#) should be called before the first use of this function.*
- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_decrypt\\_block](#) (atca\_aes\_ctr\_ctx\_t \*ctx, const uint8\_t \*ciphertext, uint8\_t \*plaintext)  
*Decrypt a block of data using CTR mode and a key within the device. [atcab\\_aes\\_ctr\\_init\(\)](#) or [atcab\\_aes\\_ctr\\_init\\_rand\(\)](#) should be called before the first use of this function.*
- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_increment](#) (atca\_aes\_ctr\_ctx\_t \*ctx)  
*Increments AES CTR counter value.*
- [ATCA\\_STATUS \\_atcab\\_exit](#) (void)
- [ATCA\\_STATUS atcab\\_wakeup](#) (void)  
*wakeup the CryptoAuth device*
- [ATCA\\_STATUS atcab\\_idle](#) (void)  
*idle the CryptoAuth device*
- [ATCA\\_STATUS atcab\\_sleep](#) (void)  
*invoke sleep on the CryptoAuth device*
- [ATCA\\_STATUS atcab\\_get\\_zone\\_size](#) (uint8\_t zone, uint16\_t slot, size\_t \*size)  
*Gets the size of the specified zone in bytes.*
- [ATCA\\_STATUS atcab\\_aes](#) (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*aes\_in, uint8\_t \*aes\_out)  
*Compute the AES-128 encrypt, decrypt, or GFM calculation.*
- [ATCA\\_STATUS atcab\\_aes\\_encrypt](#) (uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*plaintext, uint8\_t \*ciphertext)  
*Perform an AES-128 encrypt operation with a key in the device.*
- [ATCA\\_STATUS atcab\\_aes\\_encrypt\\_ext](#) (ATCADevice device, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*plaintext, uint8\_t \*ciphertext)  
*Perform an AES-128 encrypt operation with a key in the device.*
- [ATCA\\_STATUS atcab\\_aes\\_decrypt](#) (uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*ciphertext, uint8\_t \*plaintext)  
*Perform an AES-128 decrypt operation with a key in the device.*
- [ATCA\\_STATUS atcab\\_aes\\_decrypt\\_ext](#) (ATCADevice device, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*ciphertext, uint8\_t \*plaintext)  
*Perform an AES-128 decrypt operation with a key in the device.*
- [ATCA\\_STATUS atcab\\_aes\\_gfm](#) (const uint8\_t \*h, const uint8\_t \*input, uint8\_t \*output)  
*Perform a Galois Field Multiply (GFM) operation.*



- **ATCA\_STATUS atcab\_aes\_gcm\_init** (atca\_aes\_gcm\_ctx\_t \*ctx, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*iv, size\_t iv\_size)  
*Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.*
- **ATCA\_STATUS atcab\_aes\_gcm\_init\_rand** (atca\_aes\_gcm\_ctx\_t \*ctx, uint16\_t key\_id, uint8\_t key\_block, size\_t rand\_size, const uint8\_t \*free\_field, size\_t free\_field\_size, uint8\_t \*iv)  
*Initialize context for AES GCM operation with a IV composed of a random and optional fixed(free) field, which is common when starting an encrypt operation.*
- **ATCA\_STATUS atcab\_aes\_gcm\_aad\_update** (atca\_aes\_gcm\_ctx\_t \*ctx, const uint8\_t \*aad, uint32\_t aad\_size)  
*Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.*
- **ATCA\_STATUS atcab\_aes\_gcm\_encrypt\_update** (atca\_aes\_gcm\_ctx\_t \*ctx, const uint8\_t \*plaintext, uint32\_t plaintext\_size, uint8\_t \*ciphertext)  
*Encrypt data using GCM mode and a key within the ATECC608 device. atcab\_aes\_gcm\_init() or atcab\_aes\_gcm\_init\_rand() should be called before the first use of this function.*
- **ATCA\_STATUS atcab\_aes\_gcm\_encrypt\_finish** (atca\_aes\_gcm\_ctx\_t \*ctx, uint8\_t \*tag, size\_t tag\_size)  
*Complete a GCM encrypt operation returning the authentication tag.*
- **ATCA\_STATUS atcab\_aes\_gcm\_decrypt\_update** (atca\_aes\_gcm\_ctx\_t \*ctx, const uint8\_t \*ciphertext, uint32\_t ciphertext\_size, uint8\_t \*plaintext)  
*Decrypt data using GCM mode and a key within the ATECC608 device. atcab\_aes\_gcm\_init() or atcab\_aes\_gcm\_init\_rand() should be called before the first use of this function.*
- **ATCA\_STATUS atcab\_aes\_gcm\_decrypt\_finish** (atca\_aes\_gcm\_ctx\_t \*ctx, const uint8\_t \*tag, size\_t tag\_size, bool \*is\_verified)  
*Complete a GCM decrypt operation verifying the authentication tag.*
- **ATCA\_STATUS atcab\_checkmac** (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*challenge, const uint8\_t \*response, const uint8\_t \*other\_data)  
*Compares a MAC response with input values.*
- **ATCA\_STATUS atcab\_counter** (uint8\_t mode, uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Compute the Counter functions.*
- **ATCA\_STATUS atcab\_counter\_increment** (uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Increments one of the device's monotonic counters.*
- **ATCA\_STATUS atcab\_counter\_read** (uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Read one of the device's monotonic counters.*
- **ATCA\_STATUS atcab\_derivekey** (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*mac)  
*Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.*
- **ATCA\_STATUS atcab\_ecdh\_base** (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, uint8\_t \*out\_nonce)  
*Base function for generating premaster secret key using ECDH.*
- **ATCA\_STATUS atcab\_ecdh** (uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms)  
*ECDH command with a private key in a slot and the premaster secret is returned in the clear.*
- **ATCA\_STATUS atcab\_ecdh\_enc** (uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*read\_key, uint16\_t read\_key\_id, const uint8\_t num\_in[(20)])  
*ECDH command with a private key in a slot and the premaster secret is read from the next slot.*
- **ATCA\_STATUS atcab\_ecdh\_ioenc** (uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*io\_key)  
*ECDH command with a private key in a slot and the premaster secret is returned encrypted using the IO protection key.*
- **ATCA\_STATUS atcab\_ecdh\_tempkey** (const uint8\_t \*public\_key, uint8\_t \*pms)  
*ECDH command with a private key in TempKey and the premaster secret is returned in the clear.*
- **ATCA\_STATUS atcab\_ecdh\_tempkey\_ioenc** (const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*io\_key)  
*ECDH command with a private key in TempKey and the premaster secret is returned encrypted using the IO protection key.*
- **ATCA\_STATUS atcab\_gendig** (uint8\_t zone, uint16\_t key\_id, const uint8\_t \*other\_data, uint8\_t other\_data\_size)

- Issues a GenDig command, which performs a SHA256 hash on the source data indicated by zone with the contents of TempKey. See the CryptoAuth datasheet for your chip to see what the values of zone correspond to.*
- **ATCA\_STATUS atcab\_genkey\_base** (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*other\_data, uint8\_t \*public\_key)  
*Issues GenKey command, which can generate a private key, compute a public key, and/or compute a digest of a public key.*
  - **ATCA\_STATUS atcab\_genkey** (uint16\_t key\_id, uint8\_t \*public\_key)  
*Issues GenKey command, which generates a new random private key in slot/handle and returns the public key.*
  - **ATCA\_STATUS atcab\_get\_pubkey** (uint16\_t key\_id, uint8\_t \*public\_key)  
*Uses GenKey command to calculate the public key from an existing private key in a slot.*
  - **ATCA\_STATUS atcab\_hmac** (uint8\_t mode, uint16\_t key\_id, uint8\_t \*digest)  
*Issues a HMAC command, which computes an HMAC/SHA-256 digest of a key stored in the device, a challenge, and other information on the device.*
  - **ATCA\_STATUS atcab\_info\_base** (uint8\_t mode, uint16\_t param2, uint8\_t \*out\_data)  
*Issues an Info command, which return internal device information and can control GPIO and the persistent latch.*
  - **ATCA\_STATUS atcab\_info** (uint8\_t \*revision)  
*Use the Info command to get the device revision (DevRev).*
  - **ATCA\_STATUS atcab\_info\_set\_latch** (bool state)  
*Use the Info command to set the persistent latch state for an ATECC608 device.*
  - **ATCA\_STATUS atcab\_info\_get\_latch** (bool \*state)  
*Use the Info command to get the persistent latch current state for an ATECC608 device.*
  - **ATCA\_STATUS atcab\_kdf** (uint8\_t mode, uint16\_t key\_id, const uint32\_t details, const uint8\_t \*message, uint8\_t \*out\_data, uint8\_t \*out\_nonce)  
*Executes the KDF command, which derives a new key in PRF, AES, or HKDF modes.*
  - **ATCA\_STATUS atcab\_lock** (uint8\_t mode, uint16\_t summary\_crc)  
*The Lock command prevents future modifications of the Configuration and/or Data and OTP zones. If the device is so configured, then this command can be used to lock individual data slots. This command fails if the designated area is already locked.*
  - **ATCA\_STATUS atcab\_lock\_config\_zone** (void)  
*Unconditionally (no CRC required) lock the config zone.*
  - **ATCA\_STATUS atcab\_lock\_config\_zone\_crc** (uint16\_t summary\_crc)  
*Lock the config zone with summary CRC.*
  - **ATCA\_STATUS atcab\_lock\_data\_zone** (void)  
*Unconditionally (no CRC required) lock the data zone (slots and OTP). for CryptoAuth devices and lock the setup for Trust Anchor device.*
  - **ATCA\_STATUS atcab\_lock\_data\_zone\_crc** (uint16\_t summary\_crc)  
*Lock the data zone (slots and OTP) with summary CRC.*
  - **ATCA\_STATUS atcab\_lock\_data\_slot** (uint16\_t slot)  
*Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (KeyConfig.Lockable=1) (for cryptoauth devices) or Lock an individual handle in shared data element on an Trust Anchor device (for Trust Anchor devices).*
  - **ATCA\_STATUS atcab\_mac** (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*challenge, uint8\_t \*digest)  
*Executes MAC command, which computes a SHA-256 digest of a key stored in the device, a challenge, and other information on the device.*
  - **ATCA\_STATUS atcab\_nonce\_base** (uint8\_t mode, uint16\_t zero, const uint8\_t \*num\_in, uint8\_t \*rand\_out)  
*Executes Nonce command, which loads a random or fixed nonce/data into the device for use by subsequent commands.*
  - **ATCA\_STATUS atcab\_nonce** (const uint8\_t \*num\_in)  
*Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.*
  - **ATCA\_STATUS atcab\_nonce\_load** (uint8\_t target, const uint8\_t \*num\_in, uint16\_t num\_in\_size)  
*Execute a Nonce command in pass-through mode to load one of the device's internal buffers with a fixed value.*
  - **ATCA\_STATUS atcab\_nonce\_rand** (const uint8\_t \*num\_in, uint8\_t \*rand\_out)

Execute a Nonce command to generate a random nonce combining a host nonce (num\_in) and a device random number.

- **ATCA\_STATUS atcab\_challenge** (const uint8\_t \*num\_in)  
Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.
- **ATCA\_STATUS atcab\_challenge\_seed\_update** (const uint8\_t \*num\_in, uint8\_t \*rand\_out)  
Execute a Nonce command to generate a random challenge combining a host nonce (num\_in) and a device random number.
- **ATCA\_STATUS atcab\_priv\_write** (uint16\_t key\_id, const uint8\_t priv\_key[36], uint16\_t write\_key\_id, const uint8\_t write\_key[32], const uint8\_t num\_in[(20)])  
Executes PrivWrite command, to write externally generated ECC private keys into the device.
- **ATCA\_STATUS atcab\_random** (uint8\_t \*rand\_out)  
Executes Random command, which generates a 32 byte random number from the device.
- **ATCA\_STATUS atcab\_random\_ext** (ATCADevice device, uint8\_t \*rand\_out)  
Executes Random command, which generates a 32 byte random number from the device.
- **ATCA\_STATUS atcab\_read\_zone** (uint8\_t zone, uint16\_t slot, uint8\_t block, uint8\_t offset, uint8\_t \*data, uint8\_t len)  
Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.
- **ATCA\_STATUS atcab\_is\_locked** (uint8\_t zone, bool \*is\_locked)  
Executes Read command, which reads the configuration zone to see if the specified zone is locked.
- **ATCA\_STATUS atcab\_is\_config\_locked** (bool \*is\_locked)  
This function check whether configuration zone is locked or not.
- **ATCA\_STATUS atcab\_is\_data\_locked** (bool \*is\_locked)  
This function check whether data/setup zone is locked or not.
- **ATCA\_STATUS atcab\_is\_slot\_locked** (uint16\_t slot, bool \*is\_locked)  
This function check whether slot/handle is locked or not.
- **ATCA\_STATUS atcab\_read\_bytes\_zone** (uint8\_t zone, uint16\_t slot, size\_t offset, uint8\_t \*data, size\_t length)  
Used to read an arbitrary number of bytes from any zone configured for clear reads.
- **ATCA\_STATUS atcab\_read\_serial\_number** (uint8\_t \*serial\_number)  
This function returns serial number of the device.
- **ATCA\_STATUS atcab\_read\_pubkey** (uint16\_t slot, uint8\_t \*public\_key)  
Executes Read command to read an ECC P256 public key from a slot configured for clear reads.
- **ATCA\_STATUS atcab\_read\_sig** (uint16\_t slot, uint8\_t \*sig)  
Executes Read command to read a 64 byte ECDSA P256 signature from a slot configured for clear reads.
- **ATCA\_STATUS atcab\_read\_config\_zone** (uint8\_t \*config\_data)  
Executes Read command to read the complete device configuration zone.
- **ATCA\_STATUS atcab\_cmp\_config\_zone** (uint8\_t \*config\_data, bool \*same\_config)  
Compares a specified configuration zone with the configuration zone currently on the device.
- **ATCA\_STATUS atcab\_read\_enc** (uint16\_t key\_id, uint8\_t block, uint8\_t \*data, const uint8\_t \*enc\_key, const uint16\_t enc\_key\_id, const uint8\_t num\_in[(20)])  
Executes Read command on a slot configured for encrypted reads and decrypts the data to return it as plaintext.
- **ATCA\_STATUS atcab\_secureboot** (uint8\_t mode, uint16\_t param2, const uint8\_t \*digest, const uint8\_t \*signature, uint8\_t \*mac)  
Executes Secure Boot command, which provides support for secure boot of an external MCU or MPU.
- **ATCA\_STATUS atcab\_secureboot\_mac** (uint8\_t mode, const uint8\_t \*digest, const uint8\_t \*signature, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)  
Executes Secure Boot command with encrypted digest and validated MAC response using the IO protection key.
- **ATCA\_STATUS atcab\_selftest** (uint8\_t mode, uint16\_t param2, uint8\_t \*result)  
Executes the SelfTest command, which performs a test of one or more of the cryptographic engines within the ATCA ECC608 chip.
- **ATCA\_STATUS atcab\_sha\_base** (uint8\_t mode, uint16\_t length, const uint8\_t \*data\_in, uint8\_t \*data\_out, uint16\_t \*data\_out\_size)

- Executes SHA command, which computes a SHA-256 or HMAC/SHA-256 digest for general purpose use by the host system.*
- [ATCA\\_STATUS atcab\\_sha\\_start](#) (void)  
*Executes SHA command to initialize SHA-256 calculation engine.*
  - [ATCA\\_STATUS atcab\\_sha\\_update](#) (const uint8\_t \*message)  
*Executes SHA command to add 64 bytes of message data to the current context.*
  - [ATCA\\_STATUS atcab\\_sha\\_end](#) (uint8\_t \*digest, uint16\_t length, const uint8\_t \*message)  
*Executes SHA command to complete SHA-256 or HMAC/SHA-256 operation.*
  - [ATCA\\_STATUS atcab\\_sha\\_read\\_context](#) (uint8\_t \*context, uint16\_t \*context\_size)  
*Executes SHA command to read the SHA-256 context back. Only for ATECC608 with SHA-256 contexts. HMAC not supported.*
  - [ATCA\\_STATUS atcab\\_sha\\_write\\_context](#) (const uint8\_t \*context, uint16\_t context\_size)  
*Executes SHA command to write (restore) a SHA-256 context into the the device. Only supported for ATECC608 with SHA-256 contexts.*
  - [ATCA\\_STATUS atcab\\_sha](#) (uint16\_t length, const uint8\_t \*message, uint8\_t \*digest)  
*Use the SHA command to compute a SHA-256 digest.*
  - [ATCA\\_STATUS atcab\\_hw\\_sha2\\_256](#) (const uint8\_t \*data, size\_t data\_size, uint8\_t \*digest)  
*Use the SHA command to compute a SHA-256 digest.*
  - [ATCA\\_STATUS atcab\\_hw\\_sha2\\_256\\_init](#) (atca\_sha256\_ctx\_t \*ctx)  
*Initialize a SHA context for performing a hardware SHA-256 operation on a device. Note that only one SHA operation can be run at a time.*
  - [ATCA\\_STATUS atcab\\_hw\\_sha2\\_256\\_update](#) (atca\_sha256\_ctx\_t \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Add message data to a SHA context for performing a hardware SHA-256 operation on a device.*
  - [ATCA\\_STATUS atcab\\_hw\\_sha2\\_256\\_finish](#) (atca\_sha256\_ctx\_t \*ctx, uint8\_t \*digest)  
*Finish SHA-256 digest for a SHA context for performing a hardware SHA-256 operation on a device.*
  - [ATCA\\_STATUS atcab\\_sha\\_hmac\\_init](#) (atca\_hmac\_sha256\_ctx\_t \*ctx, uint16\_t key\_slot)  
*Executes SHA command to start an HMAC/SHA-256 operation.*
  - [ATCA\\_STATUS atcab\\_sha\\_hmac\\_update](#) (atca\_hmac\_sha256\_ctx\_t \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Executes SHA command to add an arbitrary amount of message data to a HMAC/SHA-256 operation.*
  - [ATCA\\_STATUS atcab\\_sha\\_hmac\\_finish](#) (atca\_hmac\_sha256\_ctx\_t \*ctx, uint8\_t \*digest, uint8\_t target)  
*Executes SHA command to complete a HMAC/SHA-256 operation.*
  - [ATCA\\_STATUS atcab\\_sha\\_hmac](#) (const uint8\_t \*data, size\_t data\_size, uint16\_t key\_slot, uint8\_t \*digest, uint8\_t target)  
*Use the SHA command to compute an HMAC/SHA-256 operation.*
  - [ATCA\\_STATUS atcab\\_sign\\_base](#) (uint8\_t mode, uint16\_t key\_id, uint8\_t \*signature)  
*Executes the Sign command, which generates a signature using the ECDSA algorithm.*
  - [ATCA\\_STATUS atcab\\_sign](#) (uint16\_t key\_id, const uint8\_t \*msg, uint8\_t \*signature)  
*Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*
  - [ATCA\\_STATUS atcab\\_sign\\_internal](#) (uint16\_t key\_id, bool is\_invalidate, bool is\_full\_sn, uint8\_t \*signature)  
*Executes Sign command to sign an internally generated message.*
  - [ATCA\\_STATUS atcab\\_updateextra](#) (uint8\_t mode, uint16\_t new\_value)  
*Executes UpdateExtra command to update the values of the two extra bytes within the Configuration zone (bytes 84 and 85).*
  - [ATCA\\_STATUS atcab\\_verify](#) (uint8\_t mode, uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*public\_key, const uint8\_t \*other\_data, uint8\_t \*mac)  
*Executes the Verify command, which takes an ECDSA [R,S] signature and verifies that it is correctly generated from a given message and public key. In all cases, the signature is an input to the command.*
  - [ATCA\\_STATUS atcab\\_verify\\_extern](#) (const uint8\_t \*message, const uint8\_t \*signature, const uint8\_t \*public\_key, bool \*is\_verified)

Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

- [ATCA\\_STATUS atcab\\_verify\\_extern\\_mac](#) (const uint8\_t \*message, const uint8\_t \*signature, const uint8\_t \*public\_key, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)

Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. This function is only available on the ATECC608.

- [ATCA\\_STATUS atcab\\_verify\\_stored](#) (const uint8\_t \*message, const uint8\_t \*signature, uint16\_t key\_id, bool \*is\_verified)

Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.

- [ATCA\\_STATUS atcab\\_verify\\_stored\\_mac](#) (const uint8\_t \*message, const uint8\_t \*signature, uint16\_t key\_id, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)

Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with a public key stored in the device. This function is only available on the ATECC608.

- [ATCA\\_STATUS atcab\\_verify\\_validate](#) (uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*other\_data, bool \*is\_verified)

Executes the Verify command in Validate mode to validate a public key stored in a slot.

- [ATCA\\_STATUS atcab\\_verify\\_invalidate](#) (uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*other\_data, bool \*is\_verified)

Executes the Verify command in Invalidate mode which invalidates a previously validated public key stored in a slot.

- [ATCA\\_STATUS atcab\\_write](#) (uint8\_t zone, uint16\_t address, const uint8\_t \*value, const uint8\_t \*mac)

Executes the Write command, which writes either one four byte word or a 32-byte block to one of the EEPROM zones on the device. Depending upon the value of the WriteConfig byte for this slot, the data may be required to be encrypted by the system prior to being sent to the device. This command cannot be used to write slots configured as ECC private keys.

- [ATCA\\_STATUS atcab\\_write\\_zone](#) (uint8\_t zone, uint16\_t slot, uint8\_t block, uint8\_t offset, const uint8\_t \*data, uint8\_t len)

Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.

- [ATCA\\_STATUS atcab\\_write\\_bytes\\_zone](#) (uint8\_t zone, uint16\_t slot, size\_t offset\_bytes, const uint8\_t \*data, size\_t length)

Executes the Write command, which writes data into the configuration, otp, or data zones with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).

- [ATCA\\_STATUS atcab\\_write\\_pubkey](#) (uint16\_t slot, const uint8\_t \*public\_key)

Uses the write command to write a public key to a slot in the proper format.

- [ATCA\\_STATUS atcab\\_write\\_config\\_zone](#) (const uint8\_t \*config\_data)

Executes the Write command, which writes the configuration zone.

- [ATCA\\_STATUS atcab\\_write\\_enc](#) (uint16\_t key\_id, uint8\_t block, const uint8\_t \*data, const uint8\_t \*enc\_key, const uint16\_t enc\_key\_id, const uint8\_t num\_in[(20)])

Executes the Write command, which performs an encrypted write of a 32 byte block into given slot.

- [ATCA\\_STATUS atcab\\_write\\_config\\_counter](#) (uint16\_t counter\_id, uint32\_t counter\_value)

Initialize one of the monotonic counters in device with a specific value.

## Variables

- `SHARED_LIB_IMPORT ATCADevice_gDevice`

### 20.4.1 Detailed Description

CryptoAuthLib Basic API methods - a simple crypto authentication API. These methods manage a global ATCADevice object behind the scenes. They also manage the wake/idle state transitions so callers don't need to.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.5 atca\_bool.h File Reference

bool define for systems that don't have it

```
#include <stdbool.h>
```

### 20.5.1 Detailed Description

bool define for systems that don't have it

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.6 atca\_cfgs.c File Reference

a set of default configurations for various ATCA devices and interfaces

```
#include <stddef.h>
#include "cryptoauthlib.h"
#include "atca_cfgs.h"
#include "atca_iface.h"
#include "atca_device.h"
```

### 20.6.1 Detailed Description

a set of default configurations for various ATCA devices and interfaces

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.7 atca\_cfgs.h File Reference

a set of default configurations for various ATCA devices and interfaces

```
#include "atca_iface.h"
```

## Variables

- [ATCAIfaceCfg cfg\\_ateccx08a\\_i2c\\_default](#)  
default configuration for an ECCx08A device on the first logical I2C bus
- [ATCAIfaceCfg cfg\\_ateccx08a\\_swi\\_default](#)  
default configuration for an ECCx08A device on the logical SWI bus over UART
- [ATCAIfaceCfg cfg\\_ateccx08a\\_kitcdc\\_default](#)  
default configuration for Kit protocol over a CDC interface
- [ATCAIfaceCfg cfg\\_ateccx08a\\_kithid\\_default](#)  
default configuration for Kit protocol over a HID interface
- [ATCAIfaceCfg cfg\\_atsha20xa\\_i2c\\_default](#)  
default configuration for a SHA204A device on the first logical I2C bus
- [ATCAIfaceCfg cfg\\_atsha20xa\\_swi\\_default](#)  
default configuration for an SHA20xA device on the logical SWI bus over UART
- [ATCAIfaceCfg cfg\\_atsha20xa\\_kitcdc\\_default](#)  
default configuration for Kit protocol over a CDC interface
- [ATCAIfaceCfg cfg\\_atsha20xa\\_kithid\\_default](#)  
default configuration for Kit protocol over a HID interface for SHA204

### 20.7.1 Detailed Description

a set of default configurations for various ATCA devices and interfaces

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.7.2 Variable Documentation

#### 20.7.2.1 cfg\_ateccx08a\_i2c\_default

[ATCAIfaceCfg](#) `cfg_ateccx08a_i2c_default`

default configuration for an ECCx08A device on the first logical I2C bus

#### 20.7.2.2 cfg\_ateccx08a\_kitcdc\_default

[ATCAIfaceCfg](#) `cfg_ateccx08a_kitcdc_default`

default configuration for Kit protocol over a CDC interface



### 20.7.2.3 `cfg_ateccx08a_kithid_default`

`ATCAIfaceCfg` `cfg_ateccx08a_kithid_default`

default configuration for Kit protocol over a HID interface

### 20.7.2.4 `cfg_ateccx08a_swi_default`

`ATCAIfaceCfg` `cfg_ateccx08a_swi_default`

default configuration for an ECCx08A device on the logical SWI bus over UART

### 20.7.2.5 `cfg_atsha20xa_i2c_default`

`ATCAIfaceCfg` `cfg_atsha20xa_i2c_default`

default configuration for a SHA204A device on the first logical I2C bus

### 20.7.2.6 `cfg_atsha20xa_kitcdc_default`

`ATCAIfaceCfg` `cfg_atsha20xa_kitcdc_default`

default configuration for Kit protocol over a CDC interface

### 20.7.2.7 `cfg_atsha20xa_kithid_default`

`ATCAIfaceCfg` `cfg_atsha20xa_kithid_default`

default configuration for Kit protocol over a HID interface for SHA204

### 20.7.2.8 `cfg_atsha20xa_swi_default`

`ATCAIfaceCfg` `cfg_atsha20xa_swi_default`

default configuration for an SHA20xA device on the logical SWI bus over UART



## 20.8 atca\_command.c File Reference

Microchip CryptoAuthentication device command builder - this is the main object that builds the command byte strings for the given device. It does not execute the command. The basic flow is to call a command method to build the command you want given the parameters and then send that byte string through the device interface.

```
#include <stdlib.h>
#include <string.h>
#include "atca_command.h"
#include "cryptoauthlib.h"
```

### Functions

- [ATCA\\_STATUS initATCACommand](#) ([ATCADeviceType](#) device\_type, [ATCACommand](#) ca\_cmd)  
*Initializer for ATCACommand.*
- [ATCACommand newATCACommand](#) ([ATCADeviceType](#) device\_type)  
*constructor for ATCACommand*
- void [deleteATCACommand](#) ([ATCACommand](#) \*ca\_cmd)  
*ATCACommand destructor.*

### 20.8.1 Detailed Description

Microchip CryptoAuthentication device command builder - this is the main object that builds the command byte strings for the given device. It does not execute the command. The basic flow is to call a command method to build the command you want given the parameters and then send that byte string through the device interface.

The primary goal of the command builder is to wrap the given parameters with the correct packet size and CRC. The caller should first fill in the parameters required in the [ATCAPacket](#) parameter given to the command. The command builder will deal with the mechanics of creating a valid packet using the parameter information.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.9 atca\_command.h File Reference

Microchip Crypto Auth device command object - this is a command builder only, it does not send the command. The result of a command method is a fully formed packet, ready to send to the ATCAIFace object to dispatch.

```
#include "atca_compiler.h"
#include "atca_status.h"
#include "atca_devtypes.h"
#include <stddef.h>
```

### Data Structures

- struct [atca\\_command](#)  
*atca\_command is the C object backing ATCACommand.*

### Typedefs

- typedef struct [atca\\_command](#) \* [ATCACommand](#)

### Functions

- [ATCA\\_STATUS](#) [initATCACommand](#) ([ATCADeviceType](#) device\_type, [ATCACommand](#) ca\_cmd)  
*Initializer for ATCACommand.*
- [ATCACommand](#) [newATCACommand](#) ([ATCADeviceType](#) device\_type)  
*constructor for ATCACommand*
- void [deleteATCACommand](#) ([ATCACommand](#) \*ca\_cmd)  
*ATCACommand destructor.*

### 20.9.1 Detailed Description

Microchip Crypto Auth device command object - this is a command builder only, it does not send the command. The result of a command method is a fully formed packet, ready to send to the ATCAIFace object to dispatch.

This command object supports the ATSHA and ATECC device family. The command list is a superset of all device commands for this family. The command object differentiates the packet contents based on specific device type within the family.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.10 atca\_compiler.h File Reference

CryptoAuthLib is meant to be portable across architectures, even non-Microchip architectures and compiler environments. This file is for isolating compiler specific macros.

### 20.10.1 Detailed Description

CryptoAuthLib is meant to be portable across architectures, even non-Microchip architectures and compiler environments. This file is for isolating compiler specific macros.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.11 atca\_crypto\_hw\_aes.h File Reference

AES CTR, CBC & CMAC structure definitions.

```
#include "cryptoauthlib.h"
```

## Data Structures

- struct [atca\\_aes\\_cbc\\_ctx](#)
- struct [atca\\_aes\\_cmac\\_ctx](#)
- struct [atca\\_aes\\_ctr\\_ctx](#)

## Typedefs

- typedef struct [atca\\_aes\\_cbc\\_ctx](#) [atca\\_aes\\_cbc\\_ctx\\_t](#)
- typedef struct [atca\\_aes\\_cmac\\_ctx](#) [atca\\_aes\\_cmac\\_ctx\\_t](#)
- typedef struct [atca\\_aes\\_ctr\\_ctx](#) [atca\\_aes\\_ctr\\_ctx\\_t](#)

### 20.11.1 Detailed Description

AES CTR, CBC & CMAC structure definitions.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.11.2 Typedef Documentation

#### 20.11.2.1 [atca\\_aes\\_cbc\\_ctx\\_t](#)

```
typedef struct atca\_aes\_cbc\_ctx atca\_aes\_cbc\_ctx\_t
```

#### 20.11.2.2 [atca\\_aes\\_cmac\\_ctx\\_t](#)

```
typedef struct atca\_aes\_cmac\_ctx atca\_aes\_cmac\_ctx\_t
```

#### 20.11.2.3 [atca\\_aes\\_ctr\\_ctx\\_t](#)

```
typedef struct atca\_aes\_ctr\_ctx atca\_aes\_ctr\_ctx\_t
```

## 20.12 [atca\\_crypto\\_hw\\_aes\\_cbc.c](#) File Reference

CryptoAuthLib Basic API methods for AES CBC mode.

```
#include "cryptoauthlib.h"  
#include "atca_crypto_hw_aes.h"
```

### Functions

- [ATCA\\_STATUS atcab\\_aes\\_cbc\\_init\\_ext](#) ([ATCADevice](#) device, [atca\\_aes\\_cbc\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block, [const uint8\\_t](#) \*iv)  
*Initialize context for AES CBC operation.*
- [ATCA\\_STATUS atcab\\_aes\\_cbc\\_init](#) ([atca\\_aes\\_cbc\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block, [const uint8\\_t](#) \*iv)  
*Initialize context for AES CBC operation.*
- [ATCA\\_STATUS atcab\\_aes\\_cbc\\_encrypt\\_block](#) ([atca\\_aes\\_cbc\\_ctx\\_t](#) \*ctx, [const uint8\\_t](#) \*plaintext, [uint8\\_t](#) \*ciphertext)  
*Encrypt a block of data using CBC mode and a key within the device. [atcab\\_aes\\_cbc\\_init\(\)](#) should be called before the first use of this function.*
- [ATCA\\_STATUS atcab\\_aes\\_cbc\\_decrypt\\_block](#) ([atca\\_aes\\_cbc\\_ctx\\_t](#) \*ctx, [const uint8\\_t](#) \*ciphertext, [uint8\\_t](#) \*plaintext)  
*Decrypt a block of data using CBC mode and a key within the device. [atcab\\_aes\\_cbc\\_init\(\)](#) should be called before the first use of this function.*

### 20.12.1 Detailed Description

CryptoAuthLib Basic API methods for AES CBC mode.

The AES command supports 128-bit AES encryption or decryption of small messages or data packets in ECB mode.

#### Note

List of devices that support this command - ATECC608A, ATECC608B, & TA100. Refer to device datasheet for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.13 atca\_crypto\_hw\_aes\_cmac.c File Reference

CryptoAuthLib Basic API methods for AES CBC\_MAC mode.

```
#include "cryptoauthlib.h"
#include "atca_crypto_hw_aes.h"
```

### Functions

- [ATCA\\_STATUS atcab\\_aes\\_cmac\\_init\\_ext](#) ([ATCADevice](#) device, [atca\\_aes\\_cmac\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block)  
*Initialize a CMAC calculation using an AES-128 key in the device.*
- [ATCA\\_STATUS atcab\\_aes\\_cmac\\_init](#) ([atca\\_aes\\_cmac\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block)  
*Initialize a CMAC calculation using an AES-128 key in the device.*
- [ATCA\\_STATUS atcab\\_aes\\_cmac\\_update](#) ([atca\\_aes\\_cmac\\_ctx\\_t](#) \*ctx, [const uint8\\_t](#) \*data, [uint32\\_t](#) data\_size)  
*Add data to an initialized CMAC calculation.*
- [ATCA\\_STATUS atcab\\_aes\\_cmac\\_finish](#) ([atca\\_aes\\_cmac\\_ctx\\_t](#) \*ctx, [uint8\\_t](#) \*cmac, [uint32\\_t](#) cmac\_size)  
*Finish a CMAC operation returning the CMAC value.*

### 20.13.1 Detailed Description

CryptoAuthLib Basic API methods for AES CBC\_MAC mode.

The AES command supports 128-bit AES encryption or decryption of small messages or data packets in ECB mode.

#### Note

List of devices that support this command - ATECC608A, ATECC608B, & TA100. Refer to device datasheet for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.14 atca\_crypto\_hw\_aes\_ctr.c File Reference

CryptoAuthLib Basic API methods for AES CTR mode.

```
#include "cryptoauthlib.h"
#include "atca_crypto_hw_aes.h"
```

### Functions

- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_init\\_ext](#) ([ATCADevice](#) device, [atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block, [uint8\\_t](#) counter\_size, [const uint8\\_t](#) \*iv)  
*Initialize context for AES CTR operation with an existing IV, which is common when start a decrypt operation.*
- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_init](#) ([atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block, [uint8\\_t](#) counter\_size, [const uint8\\_t](#) \*iv)  
*Initialize context for AES CTR operation with an existing IV, which is common when start a decrypt operation.*
- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_init\\_rand\\_ext](#) ([ATCADevice](#) device, [atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block, [uint8\\_t](#) counter\_size, [uint8\\_t](#) \*iv)  
*Initialize context for AES CTR operation with a random nonce and counter set to 0 as the IV, which is common when starting an encrypt operation.*
- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_init\\_rand](#) ([atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_id, [uint8\\_t](#) key\_block, [uint8\\_t](#) counter\_size, [uint8\\_t](#) \*iv)  
*Initialize context for AES CTR operation with a random nonce and counter set to 0 as the IV, which is common when starting an encrypt operation.*
- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_increment](#) ([atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx)  
*Increments AES CTR counter value.*
- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_block](#) ([atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, [const uint8\\_t](#) \*input, [uint8\\_t](#) \*output)  
*Process a block of data using CTR mode and a key within the device. [atcab\\_aes\\_ctr\\_init\(\)](#) or [atcab\\_aes\\_ctr\\_init\\_rand\(\)](#) should be called before the first use of this function.*
- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_encrypt\\_block](#) ([atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, [const uint8\\_t](#) \*plaintext, [uint8\\_t](#) \*ciphertext)  
*Encrypt a block of data using CTR mode and a key within the device. [atcab\\_aes\\_ctr\\_init\(\)](#) or [atcab\\_aes\\_ctr\\_init\\_rand\(\)](#) should be called before the first use of this function.*
- [ATCA\\_STATUS atcab\\_aes\\_ctr\\_decrypt\\_block](#) ([atca\\_aes\\_ctr\\_ctx\\_t](#) \*ctx, [const uint8\\_t](#) \*ciphertext, [uint8\\_t](#) \*plaintext)  
*Decrypt a block of data using CTR mode and a key within the device. [atcab\\_aes\\_ctr\\_init\(\)](#) or [atcab\\_aes\\_ctr\\_init\\_rand\(\)](#) should be called before the first use of this function.*

### 20.14.1 Detailed Description

CryptoAuthLib Basic API methods for AES CTR mode.

The AES command supports 128-bit AES encryption or decryption of small messages or data packets in ECB mode.

#### Note

List of devices that support this command - ATECC608A, ATECC608B, & TA100. Refer to device datasheet for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.15 atca\_crypto\_sw.h File Reference

Common defines for CryptoAuthLib software crypto wrappers.

```
#include <stdint.h>
#include <stdlib.h>
#include "atca_config.h"
#include "atca_status.h"
#include "mbedtls/config.h"
#include <mbedtls/cipher.h>
#include <mbedtls/md.h>
```

### Macros

- #define [ATCA\\_SHA1\\_DIGEST\\_SIZE](#) (20)
- #define [ATCA\\_SHA2\\_256\\_DIGEST\\_SIZE](#) (32)
- #define [ATCA\\_SHA2\\_256\\_BLOCK\\_SIZE](#) (64)
- #define [MBEDTLS\\_CMAC\\_C](#)

### Typedefs

- typedef mbedtls\_cipher\_context\_t [atcac\\_aes\\_cmac\\_ctx](#)
- typedef mbedtls\_md\_context\_t [atcac\\_hmac\\_sha256\\_ctx](#)
- typedef mbedtls\_cipher\_context\_t [atcac\\_aes\\_gcm\\_ctx](#)
- typedef mbedtls\_md\_context\_t [atcac\\_sha1\\_ctx](#)
- typedef mbedtls\_md\_context\_t [atcac\\_sha2\\_256\\_ctx](#)

## Functions

- [ATCA\\_STATUS atcac\\_aes\\_gcm\\_encrypt\\_start](#) ([atcac\\_aes\\_gcm\\_ctx](#) \*ctx, const uint8\_t \*key, const uint8\_t key\_len, const uint8\_t \*iv, const uint8\_t iv\_len)  
*Initialize an AES-GCM context.*
- [ATCA\\_STATUS atcac\\_aes\\_gcm\\_decrypt\\_start](#) ([atcac\\_aes\\_gcm\\_ctx](#) \*ctx, const uint8\_t \*key, const uint8\_t key\_len, const uint8\_t \*iv, const uint8\_t iv\_len)  
*Initialize an AES-GCM context for decryption.*
- [ATCA\\_STATUS atcac\\_aes\\_cmac\\_init](#) ([atcac\\_aes\\_cmac\\_ctx](#) \*ctx, const uint8\_t \*key, const uint8\_t key\_len)  
*Initialize context for performing CMAC in software.*
- [ATCA\\_STATUS atcac\\_aes\\_cmac\\_update](#) ([atcac\\_aes\\_cmac\\_ctx](#) \*ctx, const uint8\_t \*data, const size\_t data\_size)  
*Update CMAC context with input data.*
- [ATCA\\_STATUS atcac\\_aes\\_cmac\\_finish](#) ([atcac\\_aes\\_cmac\\_ctx](#) \*ctx, uint8\_t \*cmac, size\_t \*cmac\_size)  
*Finish CMAC calculation and clear the CMAC context.*
- [ATCA\\_STATUS atcac\\_aes\\_gcm\\_aad\\_update](#) ([atcac\\_aes\\_gcm\\_ctx](#) \*ctx, const uint8\_t \*aad, const size\_t aad\_len)  
*Update the GCM context with additional authentication data (AAD)*
- [ATCA\\_STATUS atcac\\_aes\\_gcm\\_encrypt\\_update](#) ([atcac\\_aes\\_gcm\\_ctx](#) \*ctx, const uint8\_t \*plaintext, const size\_t pt\_len, uint8\_t \*ciphertext, size\_t \*ct\_len)  
*Encrypt a data using the initialized context.*
- [ATCA\\_STATUS atcac\\_aes\\_gcm\\_encrypt\\_finish](#) ([atcac\\_aes\\_gcm\\_ctx](#) \*ctx, uint8\_t \*tag, size\_t tag\_len)  
*Get the AES-GCM tag and free the context.*
- [ATCA\\_STATUS atcac\\_aes\\_gcm\\_decrypt\\_update](#) ([atcac\\_aes\\_gcm\\_ctx](#) \*ctx, const uint8\_t \*ciphertext, const size\_t ct\_len, uint8\_t \*plaintext, size\_t \*pt\_len)  
*Decrypt ciphertext using the initialized context.*
- [ATCA\\_STATUS atcac\\_aes\\_gcm\\_decrypt\\_finish](#) ([atcac\\_aes\\_gcm\\_ctx](#) \*ctx, const uint8\_t \*tag, size\_t tag\_len, bool \*is\_verified)  
*Compare the AES-GCM tag and free the context.*

### 20.15.1 Detailed Description

Common defines for CryptoAuthLib software crypto wrappers.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.15.2 Macro Definition Documentation

#### 20.15.2.1 ATCA\_SHA1\_DIGEST\_SIZE

```
#define ATCA_SHA1_DIGEST_SIZE (20)
```

### 20.15.2.2 ATCA\_SHA2\_256\_BLOCK\_SIZE

```
#define ATCA_SHA2_256_BLOCK_SIZE (64)
```

### 20.15.2.3 ATCA\_SHA2\_256\_DIGEST\_SIZE

```
#define ATCA_SHA2_256_DIGEST_SIZE (32)
```

### 20.15.2.4 MBEDTLS\_CMAC\_C

```
#define MBEDTLS_CMAC_C
```

## 20.15.3 Typedef Documentation

### 20.15.3.1 atcac\_aes\_cmac\_ctx

```
typedef mbedtls_cipher_context_t atcac\_aes\_cmac\_ctx
```

### 20.15.3.2 atcac\_aes\_gcm\_ctx

```
typedef mbedtls_cipher_context_t atcac\_aes\_gcm\_ctx
```

### 20.15.3.3 atcac\_hmac\_sha256\_ctx

```
typedef mbedtls_md_context_t atcac\_hmac\_sha256\_ctx
```

### 20.15.3.4 atcac\_sha1\_ctx

```
typedef mbedtls_md_context_t atcac\_sha1\_ctx
```



### 20.15.3.5 atcac\_sha2\_256\_ctx

```
typedef mbedtls_md_context_t atcac_sha2_256_ctx
```

## 20.15.4 Function Documentation

### 20.15.4.1 atcac\_aes\_cmac\_finish()

```
ATCA_STATUS atcac_aes_cmac_finish (
    atcac_aes_cmac_ctx * ctx,
    uint8_t * cmac,
    size_t * cmac_size )
```

Finish CMAC calculation and clear the CMAC context.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.15.4.2 atcac\_aes\_cmac\_init()

```
ATCA_STATUS atcac_aes_cmac_init (
    atcac_aes_cmac_ctx * ctx,
    const uint8_t * key,
    const uint8_t key_len )
```

Initialize context for performing CMAC in software.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.15.4.3 atcac\_aes\_cmac\_update()

```
ATCA_STATUS atcac_aes_cmac_update (
    atcac_aes_cmac_ctx * ctx,
    const uint8_t * data,
    const size_t data_size )
```

Update CMAC context with input data.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.15.4.4 atcac\_aes\_gcm\_aad\_update()

```
ATCA_STATUS atcac_aes_gcm_aad_update (
    atcac_aes_gcm_ctx * ctx,
    const uint8_t * aad,
    const size_t aad_len )
```

Update the GCM context with additional authentication data (AAD)

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.15.4.5 atcac\_aes\_gcm\_decrypt\_finish()

```
ATCA_STATUS atcac_aes_gcm_decrypt_finish (
    atcac_aes_gcm_ctx * ctx,
    const uint8_t * tag,
    size_t tag_len,
    bool * is_verified )
```

Compare the AES-GCM tag and free the context.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.15.4.6 atcac\_aes\_gcm\_decrypt\_start()

```
ATCA_STATUS atcac_aes_gcm_decrypt_start (
    atcac_aes_gcm_ctx * ctx,
    const uint8_t * key,
    const uint8_t key_len,
    const uint8_t * iv,
    const uint8_t iv_len )
```

Initialize an AES-GCM context for decryption.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.15.4.7 `atcac_aes_gcm_decrypt_update()`

```
ATCA_STATUS atcac_aes_gcm_decrypt_update (
    atcac_aes_gcm_ctx * ctx,
    const uint8_t * ciphertext,
    const size_t ct_len,
    uint8_t * plaintext,
    size_t * pt_len )
```

Decrypt ciphertext using the initialized context.

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.15.4.8 `atcac_aes_gcm_encrypt_finish()`

```
ATCA_STATUS atcac_aes_gcm_encrypt_finish (
    atcac_aes_gcm_ctx * ctx,
    uint8_t * tag,
    size_t tag_len )
```

Get the AES-GCM tag and free the context.

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.15.4.9 `atcac_aes_gcm_encrypt_start()`

```
ATCA_STATUS atcac_aes_gcm_encrypt_start (
    atcac_aes_gcm_ctx * ctx,
    const uint8_t * key,
    const uint8_t key_len,
    const uint8_t * iv,
    const uint8_t iv_len )
```

Initialize an AES-GCM context.

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.16 atca\_crypto\_sw\_ecdsa.c File Reference

---

### 20.15.4.10 atcac\_aes\_gcm\_encrypt\_update()

```
ATCA_STATUS atcac_aes_gcm_encrypt_update (
    atcac_aes_gcm_ctx * ctx,
    const uint8_t * plaintext,
    const size_t pt_len,
    uint8_t * ciphertext,
    size_t * ct_len )
```

Encrypt a data using the initialized context.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.16 atca\_crypto\_sw\_ecdsa.c File Reference

API wrapper for software ECDSA verify. Currently unimplemented but could be implemented via a 3rd party library such as MicroECC.

```
#include "atca_crypto_sw_ecdsa.h"
```

### Functions

- int [atcac\\_sw\\_ecdsa\\_verify\\_p256](#) (const uint8\_t msg[(256/8)], const uint8\_t signature[((256/8) \*2)], const uint8\_t public\_key[((256/8) \*2)])

*return software generated ECDSA verification result and the function is currently not implemented*

### 20.16.1 Detailed Description

API wrapper for software ECDSA verify. Currently unimplemented but could be implemented via a 3rd party library such as MicroECC.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.17 atca\_crypto\_sw\_ecdsa.h File Reference

```
#include "atca_crypto_sw.h"
#include <stddef.h>
#include <stdint.h>
```

## Macros

- #define `ATCA_ECC_P256_FIELD_SIZE` (256 / 8)
- #define `ATCA_ECC_P256_PRIVATE_KEY_SIZE` (`ATCA_ECC_P256_FIELD_SIZE`)
- #define `ATCA_ECC_P256_PUBLIC_KEY_SIZE` (`ATCA_ECC_P256_FIELD_SIZE * 2`)
- #define `ATCA_ECC_P256_SIGNATURE_SIZE` (`ATCA_ECC_P256_FIELD_SIZE * 2`)

## Functions

- int `atcac_sw_ecdsa_verify_p256` (const uint8\_t msg[(256/8)], const uint8\_t signature[((256/8) \*2)], const uint8\_t public\_key[((256/8) \*2)])  
*return software generated ECDSA verification result and the function is currently not implemented*

### 20.17.1 Detailed Description

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.18 atca\_crypto\_sw\_rand.c File Reference

API wrapper for software random.

```
#include "atca_crypto_sw_rand.h"
```

## Functions

- int `atcac_sw_random` (uint8\_t \*data, size\_t data\_size)  
*return software generated random number and the function is currently not implemented*

### 20.18.1 Detailed Description

API wrapper for software random.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.19 atca\_crypto\_sw\_rand.h File Reference

```
#include "atca_crypto_sw.h"
#include <stddef.h>
#include <stdint.h>
```

### Functions

- int [atcac\\_sw\\_random](#) (uint8\_t \*data, size\_t data\_size)  
*return software generated random number and the function is currently not implemented*

### 20.19.1 Detailed Description

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.20 atca\_crypto\_sw\_sha1.c File Reference

Wrapper API for SHA 1 routines.

```
#include "atca_crypto_sw_sha1.h"  
#include "hashes/sha1_routines.h"
```

### Functions

- int [atcac\\_sw\\_sha1](#) (const uint8\_t \*data, size\_t data\_size, uint8\_t digest[(20)])  
*Perform SHA1 hash of data in software.*

### 20.20.1 Detailed Description

Wrapper API for SHA 1 routines.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.21 atca\_crypto\_sw\_sha1.h File Reference

Wrapper API for SHA 1 routines.

```
#include "atca_crypto_sw.h"  
#include <stddef.h>  
#include <stdint.h>
```

### Functions

- int [atcac\\_sw\\_sha1\\_init](#) ([atcac\\_sha1\\_ctx](#) \*ctx)  
*Initialize context for performing SHA1 hash in software.*
- int [atcac\\_sw\\_sha1\\_update](#) ([atcac\\_sha1\\_ctx](#) \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Add data to a SHA1 hash.*
- int [atcac\\_sw\\_sha1\\_finish](#) ([atcac\\_sha1\\_ctx](#) \*ctx, uint8\_t digest[(20)])
- int [atcac\\_sw\\_sha1](#) (const uint8\_t \*data, size\_t data\_size, uint8\_t digest[(20)])  
*Perform SHA1 hash of data in software.*

### 20.21.1 Detailed Description

Wrapper API for SHA 1 routines.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.22 atca\_crypto\_sw\_sha2.c File Reference

Wrapper API for software SHA 256 routines.

```
#include "cryptoauthlib.h"
#include "atca_crypto_sw_sha2.h"
#include "hashes/sha2_routines.h"
```

### Functions

- int [atcac\\_sw\\_sha2\\_256](#) (const uint8\_t \*data, size\_t data\_size, uint8\_t digest[(32)])  
*single call convenience function which computes Hash of given data using SHA256 software*

### 20.22.1 Detailed Description

Wrapper API for software SHA 256 routines.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.23 atca\_crypto\_sw\_sha2.h File Reference

Wrapper API for software SHA 256 routines.

```
#include "atca_crypto_sw.h"
#include <stddef.h>
#include <stdint.h>
```

## Functions

- int [atcac\\_sw\\_sha2\\_256\\_init](#) ([atcac\\_sha2\\_256\\_ctx](#) \*ctx)  
*Initialize context for performing SHA256 hash in software.*
- int [atcac\\_sw\\_sha2\\_256\\_update](#) ([atcac\\_sha2\\_256\\_ctx](#) \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Add data to a SHA256 hash.*
- int [atcac\\_sw\\_sha2\\_256\\_finish](#) ([atcac\\_sha2\\_256\\_ctx](#) \*ctx, uint8\_t digest[(32)])
- int [atcac\\_sw\\_sha2\\_256](#) (const uint8\_t \*data, size\_t data\_size, uint8\_t digest[(32)])  
*single call convenience function which computes Hash of given data using SHA256 software*
- [ATCA\\_STATUS](#) [atcac\\_sha256\\_hmac\\_init](#) ([atcac\\_hmac\\_sha256\\_ctx](#) \*ctx, const uint8\_t \*key, const uint8\_t key\_len)  
*Initialize context for performing HMAC (sha256) in software.*
- [ATCA\\_STATUS](#) [atcac\\_sha256\\_hmac\\_update](#) ([atcac\\_hmac\\_sha256\\_ctx](#) \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Update HMAC context with input data.*
- [ATCA\\_STATUS](#) [atcac\\_sha256\\_hmac\\_finish](#) ([atcac\\_hmac\\_sha256\\_ctx](#) \*ctx, uint8\_t \*digest, size\_t \*digest\_len)  
*Finish CMAC calculation and clear the HMAC context.*

### 20.23.1 Detailed Description

Wrapper API for software SHA 256 routines.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.24 atca\_debug.c File Reference

Debug/Trace for CryptoAuthLib calls.

```
#include <cryptoauthlib.h>
```

## Functions

- void [atca\\_trace\\_config](#) (FILE \*fp)
- [ATCA\\_STATUS](#) [atca\\_trace](#) ([ATCA\\_STATUS](#) status)
- [ATCA\\_STATUS](#) [atca\\_trace\\_msg](#) ([ATCA\\_STATUS](#) status, const char \*msg)

## Variables

- FILE \* [g\\_trace\\_fp](#)



## 20.24.1 Detailed Description

Debug/Trace for CryptoAuthLib calls.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.24.2 Function Documentation

### 20.24.2.1 `atca_trace()`

```
ATCA_STATUS atca_trace (
    ATCA_STATUS status )
```

### 20.24.2.2 `atca_trace_config()`

```
void atca_trace_config (
    FILE * fp )
```

### 20.24.2.3 `atca_trace_msg()`

```
ATCA_STATUS atca_trace_msg (
    ATCA_STATUS status,
    const char * msg )
```

## 20.24.3 Variable Documentation

### 20.24.3.1 `g_trace_fp`

```
FILE* g_trace_fp
```

## 20.25 `atca_debug.h` File Reference

```
#include "atca_status.h"
```

### Functions

- void [atca\\_trace\\_config](#) (FILE \*fp)
- [ATCA\\_STATUS atca\\_trace](#) ([ATCA\\_STATUS](#) status)
- [ATCA\\_STATUS atca\\_trace\\_msg](#) ([ATCA\\_STATUS](#) status, const char \*msg)

### 20.25.1 Function Documentation

#### 20.25.1.1 atca\_trace()

```
ATCA\_STATUS atca_trace (  
    ATCA\_STATUS status )
```

#### 20.25.1.2 atca\_trace\_config()

```
void atca_trace_config (  
    FILE * fp )
```

#### 20.25.1.3 atca\_trace\_msg()

```
ATCA\_STATUS atca_trace_msg (  
    ATCA\_STATUS status,  
    const char * msg )
```

## 20.26 atca\_device.c File Reference

Microchip CryptoAuth device object.

```
#include <cryptoauthlib.h>
```

### Functions

- [ATCADevice newATCADevice](#) ([ATCAIfaceCfg](#) \*cfg)  
*constructor for a Microchip CryptoAuth device*
- void [deleteATCADevice](#) ([ATCADevice](#) \*ca\_dev)  
*destructor for a device NULLs reference after object is freed*
- [ATCA\\_STATUS initATCADevice](#) ([ATCAIfaceCfg](#) \*cfg, [ATCADevice](#) ca\_dev)  
*Initializer for an Microchip CryptoAuth device.*
- [ATCACommand atGetCommands](#) ([ATCADevice](#) dev)  
*returns a reference to the ATCACommand object for the device*
- [ATCAIface atGetIFace](#) ([ATCADevice](#) dev)  
*returns a reference to the ATCAIface interface object for the device*
- [ATCA\\_STATUS releaseATCADevice](#) ([ATCADevice](#) ca\_dev)  
*Release any resources associated with the device.*

## 20.26.1 Detailed Description

Microchip CryptoAuth device object.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.27 atca\_device.h File Reference

Microchip Crypto Auth device object.

```
#include "atca_command.h"
#include "atca_iface.h"
```

### Data Structures

- struct [\\_atsha204a\\_config](#)
- struct [\\_atecc508a\\_config](#)
- struct [\\_atecc608\\_config](#)
- struct [atca\\_device](#)

[atca\\_device](#) is the C object backing ATCADevice. See the [atca\\_device.h](#) file for details on the ATCADevice methods

### Macros

- #define [ATCA\\_PACKED](#)
- #define [ATCA\\_AES\\_ENABLE\\_EN\\_SHIFT](#) (0)
- #define [ATCA\\_AES\\_ENABLE\\_EN\\_MASK](#) (0x01u << [ATCA\\_AES\\_ENABLE\\_EN\\_SHIFT](#))
- #define [ATCA\\_I2C\\_ENABLE\\_EN\\_SHIFT](#) (0)
- #define [ATCA\\_I2C\\_ENABLE\\_EN\\_MASK](#) (0x01u << [ATCA\\_I2C\\_ENABLE\\_EN\\_SHIFT](#))
- #define [ATCA\\_COUNTER\\_MATCH\\_EN\\_SHIFT](#) (0)
- #define [ATCA\\_COUNTER\\_MATCH\\_EN\\_MASK](#) (0x01u << [ATCA\\_COUNTER\\_MATCH\\_EN\\_SHIFT](#))
- #define [ATCA\\_COUNTER\\_MATCH\\_KEY\\_SHIFT](#) (4)
- #define [ATCA\\_COUNTER\\_MATCH\\_KEY\\_MASK](#) (0x0Fu << [ATCA\\_COUNTER\\_MATCH\\_KEY\\_SHIFT](#))
- #define [ATCA\\_COUNTER\\_MATCH\\_KEY\(v\)](#) ([ATCA\\_COUNTER\\_MATCH\\_KEY\\_MASK](#) & (v << [ATCA\\_COUNTER\\_MATCH\\_KEY\\_SHIFT](#)))
- #define [ATCA\\_CHIP\\_MODE\\_I2C\\_EXTRA\\_SHIFT](#) (0)
- #define [ATCA\\_CHIP\\_MODE\\_I2C\\_EXTRA\\_MASK](#) (0x01u << [ATCA\\_CHIP\\_MODE\\_I2C\\_EXTRA\\_SHIFT](#))
- #define [ATCA\\_CHIP\\_MODE\\_TTL\\_EN\\_SHIFT](#) (1)
- #define [ATCA\\_CHIP\\_MODE\\_TTL\\_EN\\_MASK](#) (0x01u << [ATCA\\_CHIP\\_MODE\\_TTL\\_EN\\_SHIFT](#))
- #define [ATCA\\_CHIP\\_MODE\\_WDG\\_LONG\\_SHIFT](#) (2)
- #define [ATCA\\_CHIP\\_MODE\\_WDG\\_LONG\\_MASK](#) (0x01u << [ATCA\\_CHIP\\_MODE\\_WDG\\_LONG\\_SHIFT](#))
- #define [ATCA\\_CHIP\\_MODE\\_CLK\\_DIV\\_SHIFT](#) (3)
- #define [ATCA\\_CHIP\\_MODE\\_CLK\\_DIV\\_MASK](#) (0x1Fu << [ATCA\\_CHIP\\_MODE\\_CLK\\_DIV\\_SHIFT](#))
- #define [ATCA\\_CHIP\\_MODE\\_CLK\\_DIV\(v\)](#) ([ATCA\\_CHIP\\_MODE\\_CLK\\_DIV\\_MASK](#) & (v << [ATCA\\_CHIP\\_MODE\\_CLK\\_DIV\\_SHIFT](#)))
- #define [ATCA\\_SLOT\\_CONFIG\\_READKEY\\_SHIFT](#) (0)
- #define [ATCA\\_SLOT\\_CONFIG\\_READKEY\\_MASK](#) (0x0Fu << [ATCA\\_SLOT\\_CONFIG\\_READKEY\\_SHIFT](#))
- #define [ATCA\\_SLOT\\_CONFIG\\_READKEY\(v\)](#) ([ATCA\\_SLOT\\_CONFIG\\_READKEY\\_MASK](#) & (v << [ATCA\\_SLOT\\_CONFIG\\_READKEY\\_SHIFT](#)))
- #define [ATCA\\_SLOT\\_CONFIG\\_NOMAC\\_SHIFT](#) (4)

- `#define ATCA_SLOT_CONFIG_NOMAC_MASK (0x01u << ATCA_SLOT_CONFIG_NOMAC_SHIFT)`
- `#define ATCA_SLOT_CONFIG_LIMITED_USE_SHIFT (5)`
- `#define ATCA_SLOT_CONFIG_LIMITED_USE_MASK (0x01u << ATCA_SLOT_CONFIG_LIMITED_USE_SHIFT)`
- `#define ATCA_SLOT_CONFIG_ENCRYPTED_READ_SHIFT (6)`
- `#define ATCA_SLOT_CONFIG_ENCRYPTED_READ_MASK (0x01u << ATCA_SLOT_CONFIG_ENCRYPTED_READ_SHIF`
- `#define ATCA_SLOT_CONFIG_IS_SECRET_SHIFT (7)`
- `#define ATCA_SLOT_CONFIG_IS_SECRET_MASK (0x01u << ATCA_SLOT_CONFIG_IS_SECRET_SHIFT)`
- `#define ATCA_SLOT_CONFIG_WRITE_KEY_SHIFT (8)`
- `#define ATCA_SLOT_CONFIG_WRITE_KEY_MASK (0x0Fu << ATCA_SLOT_CONFIG_WRITE_KEY_SHIFT)`
- `#define ATCA_SLOT_CONFIG_WRITE_KEY(v) (ATCA_SLOT_CONFIG_WRITE_KEY_MASK & (v <<`  
`ATCA_SLOT_CONFIG_WRITE_KEY_SHIFT))`
- `#define ATCA_SLOT_CONFIG_WRITE_CONFIG_SHIFT (12)`
- `#define ATCA_SLOT_CONFIG_WRITE_CONFIG_MASK (0x0Fu << ATCA_SLOT_CONFIG_WRITE_CONFIG_SHIFT)`
- `#define ATCA_SLOT_CONFIG_WRITE_CONFIG(v) (ATCA_SLOT_CONFIG_WRITE_CONFIG_MASK & (v`  
`<< ATCA_SLOT_CONFIG_WRITE_CONFIG_SHIFT))`
- `#define ATCA_SLOT_CONFIG_EXT_SIG_SHIFT (0)`
- `#define ATCA_SLOT_CONFIG_EXT_SIG_MASK (0x01u << ATCA_SLOT_CONFIG_EXT_SIG_SHIFT)`
- `#define ATCA_SLOT_CONFIG_INT_SIG_SHIFT (1)`
- `#define ATCA_SLOT_CONFIG_INT_SIG_MASK (0x01u << ATCA_SLOT_CONFIG_INT_SIG_SHIFT)`
- `#define ATCA_SLOT_CONFIG_ECDH_SHIFT (2)`
- `#define ATCA_SLOT_CONFIG_ECDH_MASK (0x01u << ATCA_SLOT_CONFIG_ECDH_SHIFT)`
- `#define ATCA_SLOT_CONFIG_WRITE_ECDH_SHIFT (3)`
- `#define ATCA_SLOT_CONFIG_WRITE_ECDH_MASK (0x01u << ATCA_SLOT_CONFIG_WRITE_ECDH_SHIFT)`
- `#define ATCA_SLOT_CONFIG_GEN_KEY_SHIFT (8)`
- `#define ATCA_SLOT_CONFIG_GEN_KEY_MASK (0x01u << ATCA_SLOT_CONFIG_GEN_KEY_SHIFT)`
- `#define ATCA_SLOT_CONFIG_PRIV_WRITE_SHIFT (9)`
- `#define ATCA_SLOT_CONFIG_PRIV_WRITE_MASK (0x01u << ATCA_SLOT_CONFIG_PRIV_WRITE_SHIFT)`
- `#define ATCA_USE_LOCK_ENABLE_SHIFT (0)`
- `#define ATCA_USE_LOCK_ENABLE_MASK (0x0Fu << ATCA_USE_LOCK_ENABLE_SHIFT)`
- `#define ATCA_USE_LOCK_KEY_SHIFT (4)`
- `#define ATCA_USE_LOCK_KEY_MASK (0x0Fu << ATCA_USE_LOCK_KEY_SHIFT)`
- `#define ATCA_VOL_KEY_PERM_SLOT_SHIFT (0)`
- `#define ATCA_VOL_KEY_PERM_SLOT_MASK (0x0Fu << ATCA_VOL_KEY_PERM_SLOT_SHIFT)`
- `#define ATCA_VOL_KEY_PERM_SLOT(v) (ATCA_VOL_KEY_PERM_SLOT_MASK & (v << ATCA_VOL_KEY_PERM_SLOT`
- `#define ATCA_VOL_KEY_PERM_EN_SHIFT (7)`
- `#define ATCA_VOL_KEY_PERM_EN_MASK (0x01u << ATCA_VOL_KEY_PERM_EN_SHIFT)`
- `#define ATCA_SECURE_BOOT_MODE_SHIFT (0)`
- `#define ATCA_SECURE_BOOT_MODE_MASK (0x03u << ATCA_SECURE_BOOT_MODE_SHIFT)`
- `#define ATCA_SECURE_BOOT_MODE(v) (ATCA_SECURE_BOOT_MODE_MASK & (v << ATCA_SECURE_BOOT_MODE`
- `#define ATCA_SECURE_BOOT_PERSIST_EN_SHIFT (3)`
- `#define ATCA_SECURE_BOOT_PERSIST_EN_MASK (0x01u << ATCA_SECURE_BOOT_PERSIST_EN_SHIFT)`
- `#define ATCA_SECURE_BOOT_RAND_NONCE_SHIFT (4)`
- `#define ATCA_SECURE_BOOT_RAND_NONCE_MASK (0x01u << ATCA_SECURE_BOOT_RAND_NONCE_SHIFT)`
- `#define ATCA_SECURE_BOOT_DIGEST_SHIFT (8)`
- `#define ATCA_SECURE_BOOT_DIGEST_MASK (0x0Fu << ATCA_SECURE_BOOT_DIGEST_SHIFT)`
- `#define ATCA_SECURE_BOOT_DIGEST(v) (ATCA_SECURE_BOOT_DIGEST_MASK & (v << ATCA_SECURE_BOOT_DIG`
- `#define ATCA_SECURE_BOOT_PUB_KEY_SHIFT (12)`
- `#define ATCA_SECURE_BOOT_PUB_KEY_MASK (0x0Fu << ATCA_SECURE_BOOT_PUB_KEY_SHIFT)`
- `#define ATCA_SECURE_BOOT_PUB_KEY(v) (ATCA_SECURE_BOOT_PUB_KEY_MASK & (v <<`  
`ATCA_SECURE_BOOT_PUB_KEY_SHIFT))`
- `#define ATCA_SLOT_LOCKED(v) ((0x01 << v) & 0xFFFFu)`
- `#define ATCA_CHIP_OPT_POST_EN_SHIFT (0)`
- `#define ATCA_CHIP_OPT_POST_EN_MASK (0x01u << ATCA_CHIP_OPT_POST_EN_SHIFT)`
- `#define ATCA_CHIP_OPT_IO_PROT_EN_SHIFT (1)`
- `#define ATCA_CHIP_OPT_IO_PROT_EN_MASK (0x01u << ATCA_CHIP_OPT_IO_PROT_EN_SHIFT)`

- `#define ATCA_CHIP_OPT_KDF_AES_EN_SHIFT (2)`
- `#define ATCA_CHIP_OPT_KDF_AES_EN_MASK (0x01u << ATCA_CHIP_OPT_KDF_AES_EN_SHIFT)`
- `#define ATCA_CHIP_OPT_ECDH_PROT_SHIFT (8)`
- `#define ATCA_CHIP_OPT_ECDH_PROT_MASK (0x03u << ATCA_CHIP_OPT_ECDH_PROT_SHIFT)`
- `#define ATCA_CHIP_OPT_ECDH_PROT(v) (ATCA_CHIP_OPT_ECDH_PROT_MASK & (v << ATCA_CHIP_OPT_ECDH_PROT_SHIFT))`
- `#define ATCA_CHIP_OPT_KDF_PROT_SHIFT (10)`
- `#define ATCA_CHIP_OPT_KDF_PROT_MASK (0x03u << ATCA_CHIP_OPT_KDF_PROT_SHIFT)`
- `#define ATCA_CHIP_OPT_KDF_PROT(v) (ATCA_CHIP_OPT_KDF_PROT_MASK & (v << ATCA_CHIP_OPT_KDF_PROT_SHIFT))`
- `#define ATCA_CHIP_OPT_IO_PROT_KEY_SHIFT (12)`
- `#define ATCA_CHIP_OPT_IO_PROT_KEY_MASK (0x0Fu << ATCA_CHIP_OPT_IO_PROT_KEY_SHIFT)`
- `#define ATCA_CHIP_OPT_IO_PROT_KEY(v) (ATCA_CHIP_OPT_IO_PROT_KEY_MASK & (v << ATCA_CHIP_OPT_IO_PROT_KEY_SHIFT))`
- `#define ATCA_KEY_CONFIG_PRIVATE_SHIFT (0)`
- `#define ATCA_KEY_CONFIG_PRIVATE_MASK (0x01u << ATCA_KEY_CONFIG_PRIVATE_SHIFT)`
- `#define ATCA_KEY_CONFIG_PUB_INFO_SHIFT (1)`
- `#define ATCA_KEY_CONFIG_PUB_INFO_MASK (0x01u << ATCA_KEY_CONFIG_PUB_INFO_SHIFT)`
- `#define ATCA_KEY_CONFIG_KEY_TYPE_SHIFT (2)`
- `#define ATCA_KEY_CONFIG_KEY_TYPE_MASK (0x07u << ATCA_KEY_CONFIG_KEY_TYPE_SHIFT)`
- `#define ATCA_KEY_CONFIG_KEY_TYPE(v) (ATCA_KEY_CONFIG_KEY_TYPE_MASK & (v << ATCA_KEY_CONFIG_KEY_TYPE_SHIFT))`
- `#define ATCA_KEY_CONFIG_LOCKABLE_SHIFT (5)`
- `#define ATCA_KEY_CONFIG_LOCKABLE_MASK (0x01u << ATCA_KEY_CONFIG_LOCKABLE_SHIFT)`
- `#define ATCA_KEY_CONFIG_REQ_RANDOM_SHIFT (6)`
- `#define ATCA_KEY_CONFIG_REQ_RANDOM_MASK (0x01u << ATCA_KEY_CONFIG_REQ_RANDOM_SHIFT)`
- `#define ATCA_KEY_CONFIG_REQ_AUTH_SHIFT (7)`
- `#define ATCA_KEY_CONFIG_REQ_AUTH_MASK (0x01u << ATCA_KEY_CONFIG_REQ_AUTH_SHIFT)`
- `#define ATCA_KEY_CONFIG_AUTH_KEY_SHIFT (8)`
- `#define ATCA_KEY_CONFIG_AUTH_KEY_MASK (0x0Fu << ATCA_KEY_CONFIG_AUTH_KEY_SHIFT)`
- `#define ATCA_KEY_CONFIG_AUTH_KEY(v) (ATCA_KEY_CONFIG_AUTH_KEY_MASK & (v << ATCA_KEY_CONFIG_AUTH_KEY_SHIFT))`
- `#define ATCA_KEY_CONFIG_PERSIST_DISABLE_SHIFT (12)`
- `#define ATCA_KEY_CONFIG_PERSIST_DISABLE_MASK (0x01u << ATCA_KEY_CONFIG_PERSIST_DISABLE_SHIFT)`
- `#define ATCA_KEY_CONFIG_RFU_SHIFT (13)`
- `#define ATCA_KEY_CONFIG_RFU_MASK (0x01u << ATCA_KEY_CONFIG_RFU_SHIFT)`
- `#define ATCA_KEY_CONFIG_X509_ID_SHIFT (14)`
- `#define ATCA_KEY_CONFIG_X509_ID_MASK (0x03u << ATCA_KEY_CONFIG_X509_ID_SHIFT)`
- `#define ATCA_KEY_CONFIG_X509_ID(v) (ATCA_KEY_CONFIG_X509_ID_MASK & (v << ATCA_KEY_CONFIG_X509_ID_SHIFT))`

## Typedefs

- `typedef struct _atsha204a_config atsha204a_config_t`
- `typedef struct _atecc508a_config atecc508a_config_t`
- `typedef struct _atecc608_config atecc608_config_t`
- `typedef struct atca_device * ATCADevice`

### Functions

- [ATCA\\_STATUS initATCADevice](#) ([ATCAIfaceCfg](#) \*cfg, [ATCADevice](#) ca\_dev)  
*Initializer for an Microchip CryptoAuth device.*
- [ATCADevice newATCADevice](#) ([ATCAIfaceCfg](#) \*cfg)  
*constructor for a Microchip CryptoAuth device*
- [ATCA\\_STATUS releaseATCADevice](#) ([ATCADevice](#) ca\_dev)  
*Release any resources associated with the device.*
- void [deleteATCADevice](#) ([ATCADevice](#) \*ca\_dev)  
*destructor for a device NULLs reference after object is freed*
- [ATCACommand atGetCommands](#) ([ATCADevice](#) dev)  
*returns a reference to the ATCACommand object for the device*
- [ATCAIface atGetIFace](#) ([ATCADevice](#) dev)  
*returns a reference to the ATCAIface interface object for the device*

### 20.27.1 Detailed Description

Microchip Crypto Auth device object.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.28 atca\_devtypes.h File Reference

Microchip Crypto Auth.

### Enumerations

- enum [ATCADeviceType](#) {  
    [ATSHA204A](#) = 0, [ATECC108A](#) = 1, [ATECC508A](#) = 2, [ATECC608A](#) = 3,  
    [ATECC608B](#) = 3, [ATECC608](#) = 3, [ATSHA206A](#) = 4, [TA100](#) = 0x10,  
    [ATCA\\_DEV\\_UNKNOWN](#) = 0x20 }  
*The supported Device type in Cryptoauthlib library.*

### 20.28.1 Detailed Description

Microchip Crypto Auth.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.29 atca\_hal.c File Reference

low-level HAL - methods used to setup indirection to physical layer interface. this level does the dirty work of abstracting the higher level ATCAIFace methods from the low-level physical interfaces. Its main goal is to keep low-level details from bleeding into the logical interface implementation.

```
#include "cryptoauthlib.h"
#include "atca_hal.h"
```

### Functions

- [ATCA\\_STATUS hal\\_iface\\_register\\_hal](#) (ATCAIFaceType iface\_type, ATCAHAL\_t \*hal, ATCAHAL\_t \*\*old)  
*Register/Replace a HAL with a.*
- [ATCA\\_STATUS hal\\_iface\\_init](#) (ATCAIFaceCfg \*cfg, ATCAHAL\_t \*\*hal)  
*Standard HAL API for ATCA to initialize a physical interface.*
- [ATCA\\_STATUS hal\\_iface\\_release](#) (ATCAIFaceType iface\_type, void \*hal\_data)  
*releases a physical interface, HAL knows how to interpret hal\_data*
- [ATCA\\_STATUS hal\\_check\\_wake](#) (const uint8\_t \*response, int response\_size)  
*Utility function for hal\_wake to check the reply.*

### 20.29.1 Detailed Description

low-level HAL - methods used to setup indirection to physical layer interface. this level does the dirty work of abstracting the higher level ATCAIFace methods from the low-level physical interfaces. Its main goal is to keep low-level details from bleeding into the logical interface implementation.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.30 atca\_hal.h File Reference

low-level HAL - methods used to setup indirection to physical layer interface

```
#include <stdlib.h>
#include "atca_config.h"
#include "atca_status.h"
#include "atca_iface.h"
#include "atca_start_config.h"
#include "atca_start_iface.h"
```

### Data Structures

- struct [ATCAHAL\\_t](#)  
*an intermediary data structure to allow the HAL layer to point the standard API functions used by the upper layers to the HAL implementation for the interface. This isolates the upper layers and loosely couples the ATCAIFace object from the physical implementation.*

## Macros

- #define `ATCA_POLLING_INIT_TIME_MSEC` 1
- #define `ATCA_POLLING_FREQUENCY_TIME_MSEC` 2
- #define `ATCA_POLLING_MAX_TIME_MSEC` 2500
- #define `hal_memset_s` `atcab_memset_s`

## Functions

- `ATCA_STATUS hal_iface_init` (`ATCAIfaceCfg *`, `ATCAHAL_t **hal`)  
*Standard HAL API for ATCA to initialize a physical interface.*
- `ATCA_STATUS hal_iface_release` (`ATCAIfaceType`, `void *hal_data`)  
*releases a physical interface, HAL knows how to interpret hal\_data*
- `ATCA_STATUS hal_check_wake` (`const uint8_t *response`, `int response_size`)  
*Utility function for hal\_wake to check the reply.*
- `void atca_delay_ms` (`uint32_t ms`)  
*Timer API for legacy implementations.*
- `void atca_delay_us` (`uint32_t delay`)  
*This function delays for a number of microseconds.*
- `void hal_rtos_delay_ms` (`uint32_t ms`)  
*Timer API implemented at the HAL level.*
- `void hal_delay_ms` (`uint32_t delay`)  
*This function delays for a number of milliseconds.*
- `void hal_delay_us` (`uint32_t delay`)  
*This function delays for a number of microseconds.*
- `ATCA_STATUS hal_create_mutex` (`void **ppMutex`, `char *pName`)  
*Optional hal interfaces.*
- `ATCA_STATUS hal_destroy_mutex` (`void *pMutex`)
- `ATCA_STATUS hal_lock_mutex` (`void *pMutex`)
- `ATCA_STATUS hal_unlock_mutex` (`void *pMutex`)
- `void * hal_malloc` (`size_t size`)
- `void hal_free` (`void *ptr`)
- `ATCA_STATUS hal_iface_register_hal` (`ATCAIfaceType` `iface_type`, `ATCAHAL_t *hal`, `ATCAHAL_t **old`)  
*Register/Replace a HAL with a.*

### 20.30.1 Detailed Description

low-level HAL - methods used to setup indirection to physical layer interface

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.31 atca\_helpers.c File Reference

Helpers to support the CryptoAuthLib Basic API methods.

```
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include "cryptoauthlib.h"
#include "atca_helpers.h"
```



## Macros

- `#define B64_IS_EQUAL (uint8_t)64`
- `#define B64_IS_INVALID (uint8_t)0xFF`

## Functions

- **ATCA\_STATUS atcab\_bin2hex** (const uint8\_t \*bin, size\_t bin\_size, char \*hex, size\_t \*hex\_size)  
*Convert a binary buffer to a hex string for easy reading.*
- **ATCA\_STATUS atcab\_reversal** (const uint8\_t \*bin, size\_t bin\_size, uint8\_t \*dest, size\_t \*dest\_size)  
*To reverse the input data.*
- **ATCA\_STATUS atcab\_bin2hex\_** (const uint8\_t \*bin, size\_t bin\_size, char \*hex, size\_t \*hex\_size, bool is\_↵  
pretty, bool is\_space, bool is\_upper)  
*Function that converts a binary buffer to a hex string suitable for easy reading.*
- **ATCA\_STATUS atcab\_hex2bin\_** (const char \*hex, size\_t hex\_size, uint8\_t \*bin, size\_t \*bin\_size, bool is\_↵  
space)
- **ATCA\_STATUS atcab\_hex2bin** (const char \*hex, size\_t hex\_size, uint8\_t \*bin, size\_t \*bin\_size)  
*Function that converts a hex string to binary buffer.*
- bool **isDigit** (char c)  
*Checks to see if a character is an ASCII representation of a digit ((c >= '0') and (c <= '9'))*
- bool **isWhiteSpace** (char c)  
*Checks to see if a character is whitespace.*
- bool **isAlpha** (char c)  
*Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))*
- bool **isHexAlpha** (char c)  
*Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))*
- bool **isHex** (char c)  
*Returns true if this character is a valid hex character or if this is whitespace (The character can be included in a valid hexstring).*
- bool **isHexDigit** (char c)  
*Returns true if this character is a valid hex character.*
- **ATCA\_STATUS packHex** (const char \*ascii\_hex, size\_t ascii\_hex\_len, char \*packed\_hex, size\_t \*packed\_↵  
\_len)  
*Remove white space from a ASCII hex string.*
- bool **isBase64** (char c, const uint8\_t \*rules)  
*Returns true if this character is a valid base 64 character or if this is whitespace (A character can be included in a valid base 64 string).*
- bool **isBase64Digit** (char c, const uint8\_t \*rules)  
*Returns true if this character is a valid base 64 character.*
- uint8\_t **base64Index** (char c, const uint8\_t \*rules)  
*Returns the base 64 index of the given character.*
- char **base64Char** (uint8\_t id, const uint8\_t \*rules)  
*Returns the base 64 character of the given index.*
- **ATCA\_STATUS atcab\_base64decode\_** (const char \*encoded, size\_t encoded\_size, uint8\_t \*data, size\_t \*data\_size, const uint8\_t \*rules)  
*Decode base64 string to data with ruleset option.*
- **ATCA\_STATUS atcab\_base64encode\_** (const uint8\_t \*data, size\_t data\_size, char \*encoded, size\_t\_↵  
\*encoded\_size, const uint8\_t \*rules)  
*Encode data as base64 string with ruleset option.*
- **ATCA\_STATUS atcab\_base64encode** (const uint8\_t \*byte\_array, size\_t array\_len, char \*encoded, size\_t \*encoded\_len)  
*Encode data as base64 string.*

- [ATCA\\_STATUS atcab\\_base64decode](#) (const char \*encoded, size\_t encoded\_len, uint8\_t \*byte\_array, size\_t array\_len)  
*Decode base64 string to data.*
- int [atcab\\_memset\\_s](#) (void \*dest, size\_t destsz, int ch, size\_t count)  
*Guaranteed to perform memory writes regardless of optimization level. Matches memset\_s signature.*

### Variables

- uint8\_t [atcab\\_b64rules\\_default](#) [4] = { '+', '/', '=', 64 }
- uint8\_t [atcab\\_b64rules\\_mime](#) [4] = { '+', '/', '=', 76 }
- uint8\_t [atcab\\_b64rules\\_ursafe](#) [4] = { '-', '\_', 0, 0 }

### 20.31.1 Detailed Description

Helpers to support the CryptoAuthLib Basic API methods.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.31.2 Macro Definition Documentation

#### 20.31.2.1 B64\_IS\_EQUAL

```
#define B64_IS_EQUAL (uint8_t) 64
```

#### 20.31.2.2 B64\_IS\_INVALID

```
#define B64_IS_INVALID (uint8_t) 0xFF
```

### 20.31.3 Function Documentation

#### 20.31.3.1 atcab\_base64decode()

```
ATCA_STATUS atcab_base64decode (  
    const char * encoded,  
    size_t encoded_len,  
    uint8_t * byte_array,  
    size_t * array_len )
```

Decode base64 string to data.

## Parameters

in	<i>encoded</i>	Base64 string to be decoded.
in	<i>encoded_len</i>	Size of the base64 string in bytes.
out	<i>byte_array</i>	Decoded data will be returned here.
in, out	<i>array_len</i>	As input, the size of the byte_array buffer. As output, the length of the decoded data.

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.31.3.2 atcab\_base64decode\_()

```
ATCA_STATUS atcab_base64decode_ (
    const char * encoded,
    size_t encoded_size,
    uint8_t * data,
    size_t * data_size,
    const uint8_t * rules )
```

Decode base64 string to data with ruleset option.

## Parameters

in	<i>encoded</i>	Base64 string to be decoded.
in	<i>encoded_size</i>	Size of the base64 string in bytes.
out	<i>data</i>	Decoded data will be returned here.
in, out	<i>data_size</i>	As input, the size of the byte_array buffer. As output, the length of the decoded data.
in	<i>rules</i>	base64 ruleset to use

## 20.31.3.3 atcab\_base64encode()

```
ATCA_STATUS atcab_base64encode (
    const uint8_t * byte_array,
    size_t array_len,
    char * encoded,
    size_t * encoded_len )
```

Encode data as base64 string.

## Parameters

in	<i>byte_array</i>	Data to be encode in base64.
in	<i>array_len</i>	Size of byte_array in bytes.
in	<i>encoded</i>	Base64 output is returned here.
in, out	<i>encoded_len</i>	As input, the size of the encoded buffer. As output, the length of the encoded base64 character string.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.31.3.4 atcab\_base64encode\_()

```
ATCA_STATUS atcab_base64encode_ (
    const uint8_t * data,
    size_t data_size,
    char * encoded,
    size_t * encoded_size,
    const uint8_t * rules )
```

Encode data as base64 string with ruleset option.

### Parameters

in	<i>data</i>	[in] The input byte array that will be converted to base 64 encoded characters
in	<i>data_size</i>	[in] The length of the byte array
in	<i>encoded</i>	[in] The output converted to base 64 encoded characters.
in, out	<i>encoded_size</i>	[inout] Input: The size of the encoded buffer, Output: The length of the encoded base 64 character string
in	<i>rules</i>	[in] ruleset to use during encoding

#### 20.31.3.5 atcab\_bin2hex()

```
ATCA_STATUS atcab_bin2hex (
    const uint8_t * bin,
    size_t bin_size,
    char * hex,
    size_t * hex_size )
```

Convert a binary buffer to a hex string for easy reading.

### Parameters

in	<i>bin</i>	Input data to convert.
in	<i>bin_size</i>	Size of data to convert.
out	<i>hex</i>	Buffer that receives hex string.
in, out	<i>hex_size</i>	As input, the size of the hex buffer. As output, the size of the output hex.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.31.3.6 atcab\_bin2hex\_()

```
ATCA_STATUS atcab_bin2hex_ (
    const uint8_t * bin,
    size_t bin_size,
    char * hex,
    size_t * hex_size,
    bool is_pretty,
    bool is_space,
    bool is_upper )
```

Function that converts a binary buffer to a hex string suitable for easy reading.

#### Parameters

in	<i>bin</i>	Input data to convert.
in	<i>bin_size</i>	Size of data to convert.
out	<i>hex</i>	Buffer that receives hex string.
in, out	<i>hex_size</i>	As input, the size of the hex buffer. As output, the size of the output hex.
in	<i>is_pretty</i>	Indicates whether new lines should be added for pretty printing.
in	<i>is_space</i>	Convert the output hex with space between it.
in	<i>is_upper</i>	Convert the output hex to upper case.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.31.3.7 atcab\_hex2bin()

```
ATCA_STATUS atcab_hex2bin (
    const char * hex,
    size_t hex_size,
    uint8_t * bin,
    size_t * bin_size )
```

Function that converts a hex string to binary buffer.

#### Parameters

in	<i>hex</i>	Input buffer to convert
in	<i>hex_size</i>	Length of buffer to convert
out	<i>bin</i>	Buffer that receives binary
in, out	<i>bin_size</i>	As input, the size of the bin buffer. As output, the size of the bin data.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.31.3.8 atcab\_hex2bin\_()

```
ATCA_STATUS atcab_hex2bin_ (
    const char * hex,
    size_t hex_size,
    uint8_t * bin,
    size_t * bin_size,
    bool is_space )
```

### 20.31.3.9 atcab\_memset\_s()

```
int atcab_memset_s (
    void * dest,
    size_t destsz,
    int ch,
    size_t count )
```

Guaranteed to perform memory writes regardless of optimization level. Matches memset\_s signature.

### 20.31.3.10 atcab\_reversal()

```
ATCA_STATUS atcab_reversal (
    const uint8_t * bin,
    size_t bin_size,
    uint8_t * dest,
    size_t * dest_size )
```

To reverse the input data.

#### Parameters

in	<i>bin</i>	Input data to reverse.
in	<i>bin_size</i>	Size of data to reverse.
out	<i>dest</i>	Buffer to store reversed binary data.
in	<i>dest_size</i>	The size of the dest buffer.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.31.3.11 base64Char()

```
char base64Char (
    uint8_t id,
    const uint8_t * rules )
```

Returns the base 64 character of the given index.

### Parameters

in	<i>id</i>	index to check
in	<i>rules</i>	base64 ruleset to use

### Returns

the base 64 character of the given index

### 20.31.3.12 base64Index()

```
uint8_t base64Index (
    char c,
    const uint8_t * rules )
```

Returns the base 64 index of the given character.

### Parameters

in	<i>c</i>	character to check
in	<i>rules</i>	base64 ruleset to use

### Returns

the base 64 index of the given character

### 20.31.3.13 isAlpha()

```
bool isAlpha (
    char c )
```

Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))

### Parameters

in	<i>c</i>	character to check
----	----------	--------------------

### Returns

True if the character is a hex



#### 20.31.3.14 isBase64()

```
bool isBase64 (
    char c,
    const uint8_t * rules )
```

Returns true if this character is a valid base 64 character or if this is whitespace (A character can be included in a valid base 64 string).

##### Parameters

in	<i>c</i>	character to check
in	<i>rules</i>	base64 ruleset to use

##### Returns

True if the character can be included in a valid base 64 string

#### 20.31.3.15 isBase64Digit()

```
bool isBase64Digit (
    char c,
    const uint8_t * rules )
```

Returns true if this character is a valid base 64 character.

##### Parameters

in	<i>c</i>	character to check
in	<i>rules</i>	base64 ruleset to use

##### Returns

True if the character can be included in a valid base 64 string

#### 20.31.3.16 isDigit()

```
bool isDigit (
    char c )
```

Checks to see if a character is an ASCII representation of a digit ((c ge '0') and (c le '9'))

##### Parameters

in	<i>c</i>	character to check
----	----------	--------------------

### Returns

True if the character is a digit

### 20.31.3.17 isHex()

```
bool isHex (
    char c )
```

Returns true if this character is a valid hex character or if this is whitespace (The character can be included in a valid hexstring).

### Parameters

in	c	character to check
----	---	--------------------

### Returns

True if the character can be included in a valid hexstring

### 20.31.3.18 isHexAlpha()

```
bool isHexAlpha (
    char c )
```

Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))

### Parameters

in	c	character to check
----	---	--------------------

### Returns

True if the character is a hex

### 20.31.3.19 isHexDigit()

```
bool isHexDigit (
    char c )
```

Returns true if this character is a valid hex character.

**Parameters**

in	c	character to check
----	---	--------------------

**Returns**

True if the character can be included in a valid hexstring

**20.31.3.20 isWhiteSpace()**

```
bool isWhiteSpace (
    char c )
```

Checks to see if a character is whitespace.

**Parameters**

in	c	character to check
----	---	--------------------

**Returns**

True if the character is whitespace

**20.31.3.21 packHex()**

```
ATCA_STATUS packHex (
    const char * ascii_hex,
    size_t ascii_hex_len,
    char * packed_hex,
    size_t * packed_len )
```

Remove white space from a ASCII hex string.

**Parameters**

in	<i>ascii_hex</i>	Initial hex string to remove white space from
in	<i>ascii_hex_len</i>	Length of the initial hex string
in	<i>packed_hex</i>	Resulting hex string without white space
in, out	<i>packed_len</i>	In: Size to packed_hex buffer Out: Number of bytes in the packed hex string

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

## 20.31.4 Variable Documentation

### 20.31.4.1 atcab\_b64rules\_default

```
uint8_t atcab_b64rules_default[4] = { '+', '/', '=', 64 }
```

### 20.31.4.2 atcab\_b64rules\_mime

```
uint8_t atcab_b64rules_mime[4] = { '+', '/', '=', 76 }
```

### 20.31.4.3 atcab\_b64rules\_urllsafe

```
uint8_t atcab_b64rules_urllsafe[4] = { '-', '_', 0, 0 }
```

## 20.32 atca\_helpers.h File Reference

Helpers to support the CryptoAuthLib Basic API methods.

```
#include "cryptoauthlib.h"
```

- `SHARED_LIB_IMPORT uint8_t atcab_b64rules_default [4]`
- `SHARED_LIB_IMPORT uint8_t atcab_b64rules_mime [4]`
- `SHARED_LIB_IMPORT uint8_t atcab_b64rules_urllsafe [4]`
- `ATCA_STATUS atcab_printbin` (uint8\_t \*binary, size\_t bin\_len, bool add\_space)
- `ATCA_STATUS atcab_bin2hex` (const uint8\_t \*bin, size\_t bin\_size, char \*hex, size\_t \*hex\_size)  
*Convert a binary buffer to a hex string for easy reading.*
- `ATCA_STATUS atcab_bin2hex_` (const uint8\_t \*bin, size\_t bin\_size, char \*hex, size\_t \*hex\_size, bool is\_↵  
pretty, bool is\_space, bool is\_upper)  
*Function that converts a binary buffer to a hex string suitable for easy reading.*
- `ATCA_STATUS atcab_hex2bin` (const char \*ascii\_hex, size\_t ascii\_hex\_len, uint8\_t \*binary, size\_t \*bin\_len)  
*Function that converts a hex string to binary buffer.*
- `ATCA_STATUS atcab_hex2bin_` (const char \*hex, size\_t hex\_size, uint8\_t \*bin, size\_t \*bin\_size, bool is\_↵  
space)
- `ATCA_STATUS atcab_printbin_sp` (uint8\_t \*binary, size\_t bin\_len)
- `ATCA_STATUS atcab_printbin_label` (const char \*label, uint8\_t \*binary, size\_t bin\_len)
- `ATCA_STATUS packHex` (const char \*ascii\_hex, size\_t ascii\_hex\_len, char \*packed\_hex, size\_t \*packed\_↵  
\_len)  
*Remove white space from a ASCII hex string.*
- bool `isDigit` (char c)  
*Checks to see if a character is an ASCII representation of a digit ((c ge '0') and (c le '9'))*

- bool [isWhiteSpace](#) (char c)  
*Checks to see if a character is whitespace.*
- bool [isAlpha](#) (char c)  
*Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))*
- bool [isHexAlpha](#) (char c)  
*Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))*
- bool [isHex](#) (char c)  
*Returns true if this character is a valid hex character or if this is whitespace (The character can be included in a valid hexstring).*
- bool [isHexDigit](#) (char c)  
*Returns true if this character is a valid hex character.*
- bool [isBase64](#) (char c, const uint8\_t \*rules)  
*Returns true if this character is a valid base 64 character or if this is whitespace (A character can be included in a valid base 64 string).*
- bool [isBase64Digit](#) (char c, const uint8\_t \*rules)  
*Returns true if this character is a valid base 64 character.*
- uint8\_t [base64Index](#) (char c, const uint8\_t \*rules)  
*Returns the base 64 index of the given character.*
- char [base64Char](#) (uint8\_t id, const uint8\_t \*rules)  
*Returns the base 64 character of the given index.*
- [ATCA\\_STATUS atcab\\_base64decode](#) (const char \*encoded, size\_t encoded\_size, uint8\_t \*data, size\_t \*data\_size, const uint8\_t \*rules)  
*Decode base64 string to data with ruleset option.*
- [ATCA\\_STATUS atcab\\_base64decode](#) (const char \*encoded, size\_t encoded\_size, uint8\_t \*data, size\_t \*data\_size)  
*Decode base64 string to data.*
- [ATCA\\_STATUS atcab\\_base64encode](#) (const uint8\_t \*data, size\_t data\_size, char \*encoded, size\_t \*encoded\_size, const uint8\_t \*rules)  
*Encode data as base64 string with ruleset option.*
- [ATCA\\_STATUS atcab\\_base64encode](#) (const uint8\_t \*data, size\_t data\_size, char \*encoded, size\_t \*encoded\_size)  
*Encode data as base64 string.*
- [ATCA\\_STATUS atcab\\_reversal](#) (const uint8\_t \*bin, size\_t bin\_size, uint8\_t \*dest, size\_t \*dest\_size)  
*To reverse the input data.*
- int [atcab\\_memset\\_s](#) (void \*dest, size\_t destsz, int ch, size\_t count)  
*Guaranteed to perform memory writes regardless of optimization level. Matches memset\_s signature.*

### 20.32.1 Detailed Description

Helpers to support the CryptoAuthLib Basic API methods.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.32.2 Function Documentation

### 20.32.2.1 atcab\_base64decode()

```
ATCA_STATUS atcab_base64decode (
    const char * encoded,
    size_t encoded_len,
    uint8_t * byte_array,
    size_t * array_len )
```

Decode base64 string to data.

#### Parameters

in	<i>encoded</i>	Base64 string to be decoded.
in	<i>encoded_len</i>	Size of the base64 string in bytes.
out	<i>byte_array</i>	Decoded data will be returned here.
in, out	<i>array_len</i>	As input, the size of the byte_array buffer. As output, the length of the decoded data.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.32.2.2 atcab\_base64decode\_()

```
ATCA_STATUS atcab_base64decode_ (
    const char * encoded,
    size_t encoded_size,
    uint8_t * data,
    size_t * data_size,
    const uint8_t * rules )
```

Decode base64 string to data with ruleset option.

#### Parameters

in	<i>encoded</i>	Base64 string to be decoded.
in	<i>encoded_size</i>	Size of the base64 string in bytes.
out	<i>data</i>	Decoded data will be returned here.
in, out	<i>data_size</i>	As input, the size of the byte_array buffer. As output, the length of the decoded data.
in	<i>rules</i>	base64 ruleset to use

### 20.32.2.3 atcab\_base64encode()

```
ATCA_STATUS atcab_base64encode (
    const uint8_t * byte_array,
    size_t array_len,
```

```
char * encoded,
size_t * encoded_len )
```

Encode data as base64 string.

#### Parameters

in	<i>byte_array</i>	Data to be encode in base64.
in	<i>array_len</i>	Size of <i>byte_array</i> in bytes.
in	<i>encoded</i>	Base64 output is returned here.
in, out	<i>encoded_len</i>	As input, the size of the encoded buffer. As output, the length of the encoded base64 character string.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.32.2.4 atcab\_base64encode\_()

```
ATCA_STATUS atcab_base64encode_ (
    const uint8_t * data,
    size_t data_size,
    char * encoded,
    size_t * encoded_size,
    const uint8_t * rules )
```

Encode data as base64 string with ruleset option.

#### 20.32.2.5 atcab\_bin2hex()

```
ATCA_STATUS atcab_bin2hex (
    const uint8_t * bin,
    size_t bin_size,
    char * hex,
    size_t * hex_size )
```

Convert a binary buffer to a hex string for easy reading.

#### Parameters

in	<i>bin</i>	Input data to convert.
in	<i>bin_size</i>	Size of data to convert.
out	<i>hex</i>	Buffer that receives hex string.
in, out	<i>hex_size</i>	As input, the size of the hex buffer. As output, the size of the output hex.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.32.2.6 atcab\_bin2hex\_()**

```
ATCA_STATUS atcab_bin2hex_ (  
    const uint8_t * bin,  
    size_t bin_size,  
    char * hex,  
    size_t * hex_size,  
    bool is_pretty,  
    bool is_space,  
    bool is_upper )
```

Function that converts a binary buffer to a hex string suitable for easy reading.

**Parameters**

in	<i>bin</i>	Input data to convert.
in	<i>bin_size</i>	Size of data to convert.
out	<i>hex</i>	Buffer that receives hex string.
in, out	<i>hex_size</i>	As input, the size of the hex buffer. As output, the size of the output hex.
in	<i>is_pretty</i>	Indicates whether new lines should be added for pretty printing.
in	<i>is_space</i>	Convert the output hex with space between it.
in	<i>is_upper</i>	Convert the output hex to upper case.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.32.2.7 atcab\_hex2bin()**

```
ATCA_STATUS atcab_hex2bin (  
    const char * hex,  
    size_t hex_size,  
    uint8_t * bin,  
    size_t * bin_size )
```

Function that converts a hex string to binary buffer.

**Parameters**

in	<i>hex</i>	Input buffer to convert
in	<i>hex_size</i>	Length of buffer to convert
out	<i>bin</i>	Buffer that receives binary
in, out	<i>bin_size</i>	As input, the size of the bin buffer. As output, the size of the bin data.



**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.32.2.8 atcab\_hex2bin\_()**

```
ATCA_STATUS atcab_hex2bin_ (
    const char * hex,
    size_t hex_size,
    uint8_t * bin,
    size_t * bin_size,
    bool is_space )
```

**20.32.2.9 atcab\_memset\_s()**

```
int atcab_memset_s (
    void * dest,
    size_t destsz,
    int ch,
    size_t count )
```

Guaranteed to perform memory writes regardless of optimization level. Matches memset\_s signature.

**20.32.2.10 atcab\_printbin\_label()**

```
ATCA_STATUS atcab_printbin_label (
    const char * label,
    uint8_t * binary,
    size_t bin_len )
```

**20.32.2.11 atcab\_printbin\_sp()**

```
ATCA_STATUS atcab_printbin_sp (
    uint8_t * binary,
    size_t bin_len )
```

**20.32.2.12 atcab\_reversal()**

```
ATCA_STATUS atcab_reversal (
    const uint8_t * bin,
    size_t bin_size,
    uint8_t * dest,
    size_t * dest_size )
```

To reverse the input data.

**Parameters**

in	<i>bin</i>	Input data to reverse.
in	<i>bin_size</i>	Size of data to reverse.
out	<i>dest</i>	Buffer to store reversed binary data.
in	<i>dest_size</i>	The size of the dest buffer.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.32.2.13 base64Char()**

```
char base64Char (
    uint8_t id,
    const uint8_t * rules )
```

Returns the base 64 character of the given index.

**Parameters**

in	<i>id</i>	index to check
in	<i>rules</i>	base64 ruleset to use

**Returns**

the base 64 character of the given index

**20.32.2.14 base64Index()**

```
uint8_t base64Index (
    char c,
    const uint8_t * rules )
```

Returns the base 64 index of the given character.

**Parameters**

in	<i>c</i>	character to check
in	<i>rules</i>	base64 ruleset to use

**Returns**

the base 64 index of the given character

### 20.32.2.15 isAlpha()

```
bool isAlpha (
    char c )
```

Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))

#### Parameters

in	<i>c</i>	character to check
----	----------	--------------------

#### Returns

True if the character is a hex

### 20.32.2.16 isBase64()

```
bool isBase64 (
    char c,
    const uint8_t * rules )
```

Returns true if this character is a valid base 64 character or if this is whitespace (A character can be included in a valid base 64 string).

#### Parameters

in	<i>c</i>	character to check
in	<i>rules</i>	base64 ruleset to use

#### Returns

True if the character can be included in a valid base 64 string

### 20.32.2.17 isBase64Digit()

```
bool isBase64Digit (
    char c,
    const uint8_t * rules )
```

Returns true if this character is a valid base 64 character.

### Parameters

in	<i>c</i>	character to check
in	<i>rules</i>	base64 ruleset to use

### Returns

True if the character can be included in a valid base 64 string

### 20.32.2.18 isDigit()

```
bool isDigit (  
    char c )
```

Checks to see if a character is an ASCII representation of a digit ((c ge '0') and (c le '9'))

### Parameters

in	<i>c</i>	character to check
----	----------	--------------------

### Returns

True if the character is a digit

### 20.32.2.19 isHex()

```
bool isHex (  
    char c )
```

Returns true if this character is a valid hex character or if this is whitespace (The character can be included in a valid hexstring).

### Parameters

in	<i>c</i>	character to check
----	----------	--------------------

### Returns

True if the character can be included in a valid hexstring

### 20.32.2.20 isHexAlpha()

```
bool isHexAlpha (
    char c )
```

Checks to see if a character is an ASCII representation of hex ((c >= 'A') and (c <= 'F')) || ((c >= 'a') and (c <= 'f'))

#### Parameters

in	c	character to check
----	---	--------------------

#### Returns

True if the character is a hex

### 20.32.2.21 isHexDigit()

```
bool isHexDigit (
    char c )
```

Returns true if this character is a valid hex character.

#### Parameters

in	c	character to check
----	---	--------------------

#### Returns

True if the character can be included in a valid hexstring

### 20.32.2.22 isWhiteSpace()

```
bool isWhiteSpace (
    char c )
```

Checks to see if a character is whitespace.

#### Parameters

in	c	character to check
----	---	--------------------

#### Returns

True if the character is whitespace

### 20.32.2.23 packHex()

```
ATCA_STATUS packHex (
    const char * ascii_hex,
    size_t ascii_hex_len,
    char * packed_hex,
    size_t * packed_len )
```

Remove white space from a ASCII hex string.

#### Parameters

in	<i>ascii_hex</i>	Initial hex string to remove white space from
in	<i>ascii_hex_len</i>	Length of the initial hex string
in	<i>packed_hex</i>	Resulting hex string without white space
in, out	<i>packed_len</i>	In: Size to packed_hex buffer Out: Number of bytes in the packed hex string

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.32.3 Variable Documentation

### 20.32.3.1 atcab\_b64rules\_default

```
SHARED_LIB_IMPORT uint8_t atcab_b64rules_default[4]
```

### 20.32.3.2 atcab\_b64rules\_mime

```
SHARED_LIB_IMPORT uint8_t atcab_b64rules_mime[4]
```

### 20.32.3.3 atcab\_b64rules\_urlsafe

```
SHARED_LIB_IMPORT uint8_t atcab_b64rules_urlsafe[4]
```

## 20.33 atca\_host.c File Reference

Host side methods to support CryptoAuth computations.

```
#include "atca_host.h"
#include "crypto/atca_crypto_sw_sha2.h"
```

### Functions

- `uint8_t * atcah_include_data` (struct `atca_include_data_in_out` \*param)  
*This function copies otp and sn data into a command buffer.*
- `ATCA_STATUS atcah_nonce` (struct `atca_nonce_in_out` \*param)  
*This function calculates host side nonce with the parameters passed.*
- `ATCA_STATUS atcah_io_decrypt` (struct `atca_io_decrypt_in_out` \*param)  
*Decrypt data that's been encrypted by the IO protection key. The ECDH and KDF commands on the ATECC608 are the only ones that support this operation.*
- `ATCA_STATUS atcah_verify_mac` (struct `atca_verify_mac_in_out_t` \*param)  
*Calculate the expected MAC on the host side for the Verify command.*
- `ATCA_STATUS atcah_secureboot_enc` (struct `atca_secureboot_enc_in_out_t` \*param)  
*Encrypts the digest for the SecureBoot command when using the encrypted digest / validating mac option.*
- `ATCA_STATUS atcah_secureboot_mac` (struct `atca_secureboot_mac_in_out_t` \*param)  
*Calculates the expected MAC returned from the SecureBoot command when verification is a success.*
- `ATCA_STATUS atcah_mac` (struct `atca_mac_in_out` \*param)  
*This function generates an SHA-256 digest (MAC) of a key, challenge, and other information.*
- `ATCA_STATUS atcah_check_mac` (struct `atca_check_mac_in_out` \*param)  
*This function performs the checkmac operation to generate client response on the host side .*
- `ATCA_STATUS atcah_hmac` (struct `atca_hmac_in_out` \*param)  
*This function generates an HMAC / SHA-256 hash of a key and other information.*
- `ATCA_STATUS atcah_gen_dig` (struct `atca_gen_dig_in_out` \*param)  
*This function combines the current TempKey with a stored value.*
- `ATCA_STATUS atcah_gen_mac` (struct `atca_gen_dig_in_out` \*param)  
*This function generates mac with session key with a plain text.*
- `ATCA_STATUS atcah_write_auth_mac` (struct `atca_write_mac_in_out` \*param)  
*This function calculates the input MAC for the Write command.*
- `ATCA_STATUS atcah_privwrite_auth_mac` (struct `atca_write_mac_in_out` \*param)  
*This function calculates the input MAC for the PrivWrite command.*
- `ATCA_STATUS atcah_derive_key` (struct `atca_derive_key_in_out` \*param)  
*This function derives a key with a key and TempKey.*
- `ATCA_STATUS atcah_derive_key_mac` (struct `atca_derive_key_mac_in_out` \*param)  
*This function calculates the input MAC for a DeriveKey command.*
- `ATCA_STATUS atcah_decrypt` (struct `atca_decrypt_in_out` \*param)  
*This function decrypts 32-byte encrypted data received with the Read command.*
- `ATCA_STATUS atcah_sha256` (int32\_t len, const uint8\_t \*message, uint8\_t \*digest)  
*This function creates a SHA256 digest on a little-endian system.*
- `ATCA_STATUS atcah_gen_key_msg` (struct `atca_gen_key_in_out` \*param)  
*Calculate the PubKey digest created by GenKey and saved to TempKey.*
- `ATCA_STATUS atcah_config_to_sign_internal` (ATCADeviceType device\_type, struct `atca_sign_internal_in_out` \*param, const uint8\_t \*config)  
*Populate the slot\_config, key\_config, and is\_slot\_locked fields in the atca\_sign\_internal\_in\_out structure from the provided config zone.*

- [ATCA\\_STATUS atcah\\_sign\\_internal\\_msg](#) ([ATCADeviceType](#) device\_type, struct [atca\\_sign\\_internal\\_in\\_out](#) \*param)  
*Builds the full message that would be signed by the Sign(Internal) command.*
- [ATCA\\_STATUS atcah\\_encode\\_counter\\_match](#) (uint32\_t counter\_value, uint8\_t \*counter\_match\_value)  
*Builds the counter match value that needs to be stored in a slot.*

### 20.33.1 Detailed Description

Host side methods to support CryptoAuth computations.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.34 atca\_host.h File Reference

Definitions and Prototypes for ATCA Utility Functions.

```
#include <stdint.h>
#include "cryptoauthlib.h"
#include "calib/calib_basic.h"
```

### Data Structures

- struct [atca\\_temp\\_key](#)  
*Structure to hold TempKey fields.*
- struct [atca\\_include\\_data\\_in\\_out](#)  
*Input / output parameters for function atca\_include\_data().*
- struct [atca\\_nonce\\_in\\_out](#)  
*Input/output parameters for function atca\_nonce().*
- struct [atca\\_io\\_decrypt\\_in\\_out](#)
- struct [atca\\_verify\\_mac](#)
- struct [atca\\_secureboot\\_enc\\_in\\_out](#)
- struct [atca\\_secureboot\\_mac\\_in\\_out](#)
- struct [atca\\_mac\\_in\\_out](#)  
*Input/output parameters for function atca\_mac().*
- struct [atca\\_hmac\\_in\\_out](#)  
*Input/output parameters for function atca\_hmac().*
- struct [atca\\_gen\\_dig\\_in\\_out](#)  
*Input/output parameters for function atcah\_gen\_dig().*
- struct [atca\\_write\\_mac\\_in\\_out](#)  
*Input/output parameters for function atcah\_write\_auth\_mac() and atcah\_privwrite\_auth\_mac().*
- struct [atca\\_derive\\_key\\_in\\_out](#)  
*Input/output parameters for function atcah\_derive\_key().*
- struct [atca\\_derive\\_key\\_mac\\_in\\_out](#)  
*Input/output parameters for function atcah\_derive\_key\_mac().*
- struct [atca\\_decrypt\\_in\\_out](#)



- *Input/output parameters for function `atca_decrypt()`.*
- struct `atca_check_mac_in_out`  
*Input/output parameters for function `atcah_check_mac()`.*
- struct `atca_verify_in_out`  
*Input/output parameters for function `atcah_verify()`.*
- struct `atca_gen_key_in_out`  
*Input/output parameters for calculating the PubKey digest put into TempKey by the GenKey command with the `atcah_gen_key_msg()` function.*
- struct `atca_sign_internal_in_out`  
*Input/output parameters for calculating the message and digest used by the Sign(internal) command. Used with the `atcah_sign_internal_msg()` function.*

## Macros

### Definitions for ATECC Message Sizes to Calculate a SHA256 Hash

"||" is the concatenation operator. The number in braces is the length of the hash input value in bytes.

- #define `ATCA_MSG_SIZE_NONCE` (55)  
*RandOut{32} || NumIn{20} || OpCode{1} || Mode{1} || LSB of Param2{1}.*
- #define `ATCA_MSG_SIZE_MAC` (88)  
*(Key or TempKey){32} || (Challenge or TempKey){32} || OpCode{1} || Mode{1} || Param2{2} || (OTP0\_7 or 0){8} || (OTP8\_10 or 0){3} || SN8{1} || (SN4\_7 or 0){4} || SN0\_1{2} || (SN2\_3 or 0){2}*
- #define `ATCA_MSG_SIZE_HMAC` (88)
- #define `ATCA_MSG_SIZE_GEN_DIG` (96)  
*KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0\_1{2} || 0{25} || TempKey{32}.*
- #define `ATCA_MSG_SIZE_DERIVE_KEY` (96)  
*KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0\_1{2} || 0{25} || TempKey{32}.*
- #define `ATCA_MSG_SIZE_DERIVE_KEY_MAC` (39)  
*KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0\_1{2}.*
- #define `ATCA_MSG_SIZE_ENCRYPT_MAC` (96)  
*KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0\_1{2} || 0{25} || TempKey{32}.*
- #define `ATCA_MSG_SIZE_PRIVWRITE_MAC` (96)  
*KeyId{32} || OpCode{1} || Param1{1} || Param2{2} || SN8{1} || SN0\_1{2} || 0{21} || PlainText{36}.*
- #define `ATCA_COMMAND_HEADER_SIZE` ( 4)
- #define `ATCA_GENDIG_ZEROS_SIZE` (25)
- #define `ATCA_WRITE_MAC_ZEROS_SIZE` (25)
- #define `ATCA_PRIVWRITE_MAC_ZEROS_SIZE` (21)
- #define `ATCA_PRIVWRITE_PLAIN_TEXT_SIZE` (36)
- #define `ATCA_DERIVE_KEY_ZEROS_SIZE` (25)
- #define `ATCA_HMAC_BLOCK_SIZE` (64)
- #define `ENCRYPTION_KEY_SIZE` (64)

### Default Fixed Byte Values of Serial Number (SN[0:1] and SN[8])

- #define `ATCA_SN_0_DEF` (0x01)
- #define `ATCA_SN_1_DEF` (0x23)
- #define `ATCA_SN_8_DEF` (0xEE)

### Definition for TempKey Mode

- #define `MAC_MODE_USE_TEMPKEY_MASK` ((uint8\_t)0x03)  
*mode mask for MAC command when using TempKey*

## Typedefs

- typedef struct [atca\\_temp\\_key](#) [atca\\_temp\\_key\\_t](#)  
*Structure to hold TempKey fields.*
- typedef struct [atca\\_nonce\\_in\\_out](#) [atca\\_nonce\\_in\\_out\\_t](#)
- typedef struct [atca\\_io\\_decrypt\\_in\\_out](#) [atca\\_io\\_decrypt\\_in\\_out\\_t](#)
- typedef struct [atca\\_verify\\_mac](#) [atca\\_verify\\_mac\\_in\\_out\\_t](#)
- typedef struct [atca\\_secureboot\\_enc\\_in\\_out](#) [atca\\_secureboot\\_enc\\_in\\_out\\_t](#)
- typedef struct [atca\\_secureboot\\_mac\\_in\\_out](#) [atca\\_secureboot\\_mac\\_in\\_out\\_t](#)
- typedef struct [atca\\_mac\\_in\\_out](#) [atca\\_mac\\_in\\_out\\_t](#)
- typedef struct [atca\\_gen\\_dig\\_in\\_out](#) [atca\\_gen\\_dig\\_in\\_out\\_t](#)  
*Input/output parameters for function [atcah\\_gen\\_dig\(\)](#).*
- typedef struct [atca\\_write\\_mac\\_in\\_out](#) [atca\\_write\\_mac\\_in\\_out\\_t](#)  
*Input/output parameters for function [atcah\\_write\\_auth\\_mac\(\)](#) and [atcah\\_privwrite\\_auth\\_mac\(\)](#).*
- typedef struct [atca\\_check\\_mac\\_in\\_out](#) [atca\\_check\\_mac\\_in\\_out\\_t](#)  
*Input/output parameters for function [atcah\\_check\\_mac\(\)](#).*
- typedef struct [atca\\_verify\\_in\\_out](#) [atca\\_verify\\_in\\_out\\_t](#)
- typedef struct [atca\\_gen\\_key\\_in\\_out](#) [atca\\_gen\\_key\\_in\\_out\\_t](#)  
*Input/output parameters for calculating the PubKey digest put into TempKey by the GenKey command with the [atcah\\_gen\\_key\\_msg\(\)](#) function.*
- typedef struct [atca\\_sign\\_internal\\_in\\_out](#) [atca\\_sign\\_internal\\_in\\_out\\_t](#)  
*Input/output parameters for calculating the message and digest used by the Sign(internal) command. Used with the [atcah\\_sign\\_internal\\_msg\(\)](#) function.*

## Functions

- [ATCA\\_STATUS atcah\\_nonce](#) (struct [atca\\_nonce\\_in\\_out](#) \*param)  
*This function calculates host side nonce with the parameters passed.*
- [ATCA\\_STATUS atcah\\_mac](#) (struct [atca\\_mac\\_in\\_out](#) \*param)  
*This function generates an SHA-256 digest (MAC) of a key, challenge, and other information.*
- [ATCA\\_STATUS atcah\\_check\\_mac](#) (struct [atca\\_check\\_mac\\_in\\_out](#) \*param)  
*This function performs the checkmac operation to generate client response on the host side .*
- [ATCA\\_STATUS atcah\\_hmac](#) (struct [atca\\_hmac\\_in\\_out](#) \*param)  
*This function generates an HMAC / SHA-256 hash of a key and other information.*
- [ATCA\\_STATUS atcah\\_gen\\_dig](#) (struct [atca\\_gen\\_dig\\_in\\_out](#) \*param)  
*This function combines the current TempKey with a stored value.*
- [ATCA\\_STATUS atcah\\_gen\\_mac](#) (struct [atca\\_gen\\_dig\\_in\\_out](#) \*param)  
*This function generates mac with session key with a plain text.*
- [ATCA\\_STATUS atcah\\_write\\_auth\\_mac](#) (struct [atca\\_write\\_mac\\_in\\_out](#) \*param)  
*This function calculates the input MAC for the Write command.*
- [ATCA\\_STATUS atcah\\_privwrite\\_auth\\_mac](#) (struct [atca\\_write\\_mac\\_in\\_out](#) \*param)  
*This function calculates the input MAC for the PrivWrite command.*
- [ATCA\\_STATUS atcah\\_derive\\_key](#) (struct [atca\\_derive\\_key\\_in\\_out](#) \*param)  
*This function derives a key with a key and TempKey.*
- [ATCA\\_STATUS atcah\\_derive\\_key\\_mac](#) (struct [atca\\_derive\\_key\\_mac\\_in\\_out](#) \*param)  
*This function calculates the input MAC for a DeriveKey command.*
- [ATCA\\_STATUS atcah\\_decrypt](#) (struct [atca\\_decrypt\\_in\\_out](#) \*param)  
*This function decrypts 32-byte encrypted data received with the Read command.*
- [ATCA\\_STATUS atcah\\_sha256](#) (int32\_t len, const uint8\_t \*message, uint8\_t \*digest)  
*This function creates a SHA256 digest on a little-endian system.*
- uint8\_t \* [atcah\\_include\\_data](#) (struct [atca\\_include\\_data\\_in\\_out](#) \*param)

*This function copies otp and sn data into a command buffer.*

- [ATCA\\_STATUS atcah\\_gen\\_key\\_msg](#) (struct [atca\\_gen\\_key\\_in\\_out](#) \*param)

*Calculate the PubKey digest created by GenKey and saved to TempKey.*

- [ATCA\\_STATUS atcah\\_config\\_to\\_sign\\_internal](#) ([ATCADeviceType](#) device\_type, struct [atca\\_sign\\_internal\\_in\\_out](#) \*param, const uint8\_t \*config)

*Populate the slot\_config, key\_config, and is\_slot\_locked fields in the [atca\\_sign\\_internal\\_in\\_out](#) structure from the provided config zone.*

- [ATCA\\_STATUS atcah\\_sign\\_internal\\_msg](#) ([ATCADeviceType](#) device\_type, struct [atca\\_sign\\_internal\\_in\\_out](#) \*param)

*Builds the full message that would be signed by the Sign(Internal) command.*

- [ATCA\\_STATUS atcah\\_verify\\_mac](#) ([atca\\_verify\\_mac\\_in\\_out\\_t](#) \*param)

*Calculate the expected MAC on the host side for the Verify command.*

- [ATCA\\_STATUS atcah\\_secureboot\\_enc](#) ([atca\\_secureboot\\_enc\\_in\\_out\\_t](#) \*param)

*Encrypts the digest for the SecureBoot command when using the encrypted digest / validating mac option.*

- [ATCA\\_STATUS atcah\\_secureboot\\_mac](#) ([atca\\_secureboot\\_mac\\_in\\_out\\_t](#) \*param)

*Calculates the expected MAC returned from the SecureBoot command when verification is a success.*

- [ATCA\\_STATUS atcah\\_encode\\_counter\\_match](#) (uint32\_t counter, uint8\_t \*counter\_match)

*Builds the counter match value that needs to be stored in a slot.*

- [ATCA\\_STATUS atcah\\_io\\_decrypt](#) (struct [atca\\_io\\_decrypt\\_in\\_out](#) \*param)

*Decrypt data that's been encrypted by the IO protection key. The ECDH and KDF commands on the ATECC608 are the only ones that support this operation.*

### 20.34.1 Detailed Description

Definitions and Prototypes for ATCA Utility Functions.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.35 atca\_iface.c File Reference

Microchip CryptoAuthLib hardware interface object.

```
#include <stdlib.h>
#include "atca_iface.h"
#include "hal/atca_hal.h"
#include "atca_config.h"
```

### Functions

- [ATCA\\_STATUS initATCAiface](#) ([ATCAifaceCfg](#) \*cfg, [ATCAiface](#) ca\_iface)

*Initializer for ATCAiface objects.*

- [ATCAiface newATCAiface](#) ([ATCAifaceCfg](#) \*cfg)

*Constructor for ATCAiface objects.*

- [ATCA\\_STATUS atinit](#) ([ATCAiface](#) ca\_iface)

*Performs the HAL initialization by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_init\(\)](#) function should be called instead.*

- [ATCA\\_STATUS atsend](#) ([ATCAIface](#) ca\_iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*Sends the data to the device by calling intermediate HAL wrapper function.*
- [ATCA\\_STATUS atreceive](#) ([ATCAIface](#) ca\_iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)  
*Receives data from the device by calling intermediate HAL wrapper function.*
- [ATCA\\_STATUS atwake](#) ([ATCAIface](#) ca\_iface)  
*Wakes up the device by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_wakeup\(\)](#) function should be used instead.*
- [ATCA\\_STATUS atidle](#) ([ATCAIface](#) ca\_iface)  
*Puts the device into idle state by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_idle\(\)](#) function should be used instead.*
- [ATCA\\_STATUS atsleep](#) ([ATCAIface](#) ca\_iface)  
*Puts the device into sleep state by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_sleep\(\)](#) function should be used instead.*
- [ATCAIfaceCfg \\* atgetifacecfg](#) ([ATCAIface](#) ca\_iface)  
*Returns the logical interface configuration for the device.*
- void \* [atgetifacehaldat](#) ([ATCAIface](#) ca\_iface)  
*Returns the HAL data pointer for the device.*
- [ATCA\\_STATUS releaseATCAIface](#) ([ATCAIface](#) ca\_iface)  
*Instruct the HAL driver to release any resources associated with this interface.*
- void [deleteATCAIface](#) ([ATCAIface](#) \*ca\_iface)  
*Instruct the HAL driver to release any resources associated with this interface, then delete the object.*

### 20.35.1 Detailed Description

Microchip CryptoAuthLib hardware interface object.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.36 atca\_iface.h File Reference

Microchip Crypto Auth hardware interface object.

```
#include "atca_devtypes.h"
#include <stdint.h>
#include "atca_status.h"
```

### Data Structures

- struct [ATCAIfaceCfg](#)
- struct [atca\\_iface](#)  
*[atca\\_iface](#) is the C object backing ATCAIface. See the [atca\\_iface.h](#) file for details on the ATCAIface methods*

### Typedefs

- typedef struct [atca\\_iface](#) \* [ATCAIface](#)

## Enumerations

- enum [ATCAIfaceType](#) {  
[ATCA\\_I2C\\_IFACE](#), [ATCA\\_SWI\\_IFACE](#), [ATCA\\_UART\\_IFACE](#), [ATCA\\_SPI\\_IFACE](#),  
[ATCA\\_HID\\_IFACE](#), [ATCA\\_CUSTOM\\_IFACE](#), [ATCA\\_UNKNOWN\\_IFACE](#) }
- enum [ATCAKitType](#) {  
[ATCA\\_KIT\\_AUTO\\_IFACE](#), [ATCA\\_KIT\\_I2C\\_IFACE](#), [ATCA\\_KIT\\_SWI\\_IFACE](#), [ATCA\\_KIT\\_SPI\\_IFACE](#),  
[ATCA\\_KIT\\_UNKNOWN\\_IFACE](#) }

## Functions

- [ATCA\\_STATUS](#) [initATCAIface](#) ([ATCAIfaceCfg](#) \*cfg, [ATCAIface](#) ca\_iface)  
*Initializer for ATCAIface objects.*
- [ATCAIface](#) [newATCAIface](#) ([ATCAIfaceCfg](#) \*cfg)  
*Constructor for ATCAIface objects.*
- [ATCA\\_STATUS](#) [releaseATCAIface](#) ([ATCAIface](#) ca\_iface)  
*Instruct the HAL driver to release any resources associated with this interface.*
- void [deleteATCAIface](#) ([ATCAIface](#) \*ca\_iface)  
*Instruct the HAL driver to release any resources associated with this interface, then delete the object.*
- [ATCA\\_STATUS](#) [atinit](#) ([ATCAIface](#) ca\_iface)  
*Performs the HAL initialization by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_init\(\)](#) function should be called instead.*
- [ATCA\\_STATUS](#) [atpostinit](#) ([ATCAIface](#) ca\_iface)
- [ATCA\\_STATUS](#) [atsend](#) ([ATCAIface](#) ca\_iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*Sends the data to the device by calling intermediate HAL wrapper function.*
- [ATCA\\_STATUS](#) [atreceive](#) ([ATCAIface](#) ca\_iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)  
*Receives data from the device by calling intermediate HAL wrapper function.*
- [ATCA\\_STATUS](#) [atwake](#) ([ATCAIface](#) ca\_iface)  
*Wakes up the device by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_wakeup\(\)](#) function should be used instead.*
- [ATCA\\_STATUS](#) [atidle](#) ([ATCAIface](#) ca\_iface)  
*Puts the device into idle state by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_idle\(\)](#) function should be used instead.*
- [ATCA\\_STATUS](#) [atsleep](#) ([ATCAIface](#) ca\_iface)  
*Puts the device into sleep state by calling intermediate HAL wrapper function. If using the basic API, the [atcab\\_sleep\(\)](#) function should be used instead.*
- [ATCAIfaceCfg](#) \* [atgetifacecfg](#) ([ATCAIface](#) ca\_iface)  
*Returns the logical interface configuration for the device.*
- void \* [atgetifacehaldat](#) ([ATCAIface](#) ca\_iface)  
*Returns the HAL data pointer for the device.*

### 20.36.1 Detailed Description

Microchip Crypto Auth hardware interface object.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.37 atca\_jwt.c File Reference

Utilities to create and verify a JSON Web Token (JWT)

```
#include "cryptoauthlib.h"
#include "atca_helpers.h"
#include "crypto/atca_crypto_sw_sha2.h"
#include "jwt/atca_jwt.h"
#include <stdio.h>
```

### Functions

- void [atca\\_jwt\\_check\\_payload\\_start](#) ([atca\\_jwt\\_t](#) \*jwt)  
*Check the provided context to see what character needs to be added in order to append a claim.*
- [ATCA\\_STATUS](#) [atca\\_jwt\\_init](#) ([atca\\_jwt\\_t](#) \*jwt, char \*buf, [uint16\\_t](#) buflen)  
*Initialize a JWT structure.*
- [ATCA\\_STATUS](#) [atca\\_jwt\\_finalize](#) ([atca\\_jwt\\_t](#) \*jwt, [uint16\\_t](#) key\_id)  
*Close the claims of a token, encode them, then sign the result.*
- [ATCA\\_STATUS](#) [atca\\_jwt\\_add\\_claim\\_string](#) ([atca\\_jwt\\_t](#) \*jwt, const char \*claim, const char \*value)  
*Add a string claim to a token.*
- [ATCA\\_STATUS](#) [atca\\_jwt\\_add\\_claim\\_numeric](#) ([atca\\_jwt\\_t](#) \*jwt, const char \*claim, [int32\\_t](#) value)  
*Add a numeric claim to a token.*
- [ATCA\\_STATUS](#) [atca\\_jwt\\_verify](#) (const char \*buf, [uint16\\_t](#) buflen, const [uint8\\_t](#) \*pubkey)  
*Verifies the signature of a jwt using the provided public key.*

### 20.37.1 Detailed Description

Utilities to create and verify a JSON Web Token (JWT)

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.38 atca\_jwt.h File Reference

Utilities to create and verify a JSON Web Token (JWT)

```
#include "cryptoauthlib.h"
```

### Data Structures

- struct [atca\\_jwt\\_t](#)  
*Structure to hold metadata information about the jwt being built.*

## Functions

- [ATCA\\_STATUS atca\\_jwt\\_init](#) ([atca\\_jwt\\_t](#) \*jwt, char \*buf, uint16\_t buflen)  
*Initialize a JWT structure.*
- [ATCA\\_STATUS atca\\_jwt\\_add\\_claim\\_string](#) ([atca\\_jwt\\_t](#) \*jwt, const char \*claim, const char \*value)  
*Add a string claim to a token.*
- [ATCA\\_STATUS atca\\_jwt\\_add\\_claim\\_numeric](#) ([atca\\_jwt\\_t](#) \*jwt, const char \*claim, int32\_t value)  
*Add a numeric claim to a token.*
- [ATCA\\_STATUS atca\\_jwt\\_finalize](#) ([atca\\_jwt\\_t](#) \*jwt, uint16\_t key\_id)  
*Close the claims of a token, encode them, then sign the result.*
- void [atca\\_jwt\\_check\\_payload\\_start](#) ([atca\\_jwt\\_t](#) \*jwt)  
*Check the provided context to see what character needs to be added in order to append a claim.*
- [ATCA\\_STATUS atca\\_jwt\\_verify](#) (const char \*buf, uint16\_t buflen, const uint8\_t \*pubkey)  
*Verifies the signature of a jwt using the provided public key.*

### 20.38.1 Detailed Description

Utilities to create and verify a JSON Web Token (JWT)

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.39 atca\_mbedtls\_ecdh.c File Reference

```
#include "mbedtls/config.h"
```

## 20.40 atca\_mbedtls\_ecdsa.c File Reference

```
#include "mbedtls/config.h"
```

## 20.41 atca\_mbedtls\_wrap.c File Reference

Wrapper functions to replace cryptoauthlib software crypto functions with the mbedtls equivalent.

```
#include "mbedtls/config.h"
#include <stdlib.h>
#include "mbedtls/cmac.h"
#include "mbedtls/pk.h"
#include "mbedtls/ecp.h"
#include "mbedtls/x509_crt.h"
#include "cryptoauthlib.h"
#include "crypto/atca_crypto_sw.h"
#include "atcacert/atcacert_client.h"
#include "atcacert/atcacert_def.h"
```

## Macros

- #define `mbedtls_calloc` `calloc`
- #define `mbedtls_free` `free`

## Functions

- **ATCA\_STATUS** `atcac_aes_gcm_aad_update` (`atcac_aes_gcm_ctx` \*ctx, const uint8\_t \*aad, const size\_t aad\_len)  
*Update the GCM context with additional authentication data (AAD)*
- **ATCA\_STATUS** `atcac_aes_gcm_encrypt_start` (`atcac_aes_gcm_ctx` \*ctx, const uint8\_t \*key, const uint8\_t key\_len, const uint8\_t \*iv, const uint8\_t iv\_len)  
*Initialize an AES-GCM context.*
- **ATCA\_STATUS** `atcac_aes_gcm_encrypt_update` (`atcac_aes_gcm_ctx` \*ctx, const uint8\_t \*plaintext, const size\_t pt\_len, uint8\_t \*ciphertext, size\_t \*ct\_len)  
*Encrypt a data using the initialized context.*
- **ATCA\_STATUS** `atcac_aes_gcm_encrypt_finish` (`atcac_aes_gcm_ctx` \*ctx, uint8\_t \*tag, size\_t tag\_len)  
*Get the AES-GCM tag and free the context.*
- **ATCA\_STATUS** `atcac_aes_gcm_decrypt_start` (`atcac_aes_gcm_ctx` \*ctx, const uint8\_t \*key, const uint8\_t key\_len, const uint8\_t \*iv, const uint8\_t iv\_len)  
*Initialize an AES-GCM context for decryption.*
- **ATCA\_STATUS** `atcac_aes_gcm_decrypt_update` (`atcac_aes_gcm_ctx` \*ctx, const uint8\_t \*ciphertext, const size\_t ct\_len, uint8\_t \*plaintext, size\_t \*pt\_len)  
*Decrypt ciphertext using the initialized context.*
- **ATCA\_STATUS** `atcac_aes_gcm_decrypt_finish` (`atcac_aes_gcm_ctx` \*ctx, const uint8\_t \*tag, size\_t tag\_len, bool \*is\_verified)  
*Compare the AES-GCM tag and free the context.*
- int `atcac_sw_sha1_init` (`atcac_sha1_ctx` \*ctx)  
*Initialize context for performing SHA1 hash in software.*
- int `atcac_sw_sha1_update` (`atcac_sha1_ctx` \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Add data to a SHA1 hash.*
- int `atcac_sw_sha1_finish` (`atcac_sha1_ctx` \*ctx, uint8\_t digest[**ATCA\_SHA1\_DIGEST\_SIZE**])  
*Complete the SHA1 hash in software and return the digest.*
- int `atcac_sw_sha2_256_init` (`atcac_sha2_256_ctx` \*ctx)  
*Initialize context for performing SHA256 hash in software.*
- int `atcac_sw_sha2_256_update` (`atcac_sha2_256_ctx` \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Add data to a SHA256 hash.*
- int `atcac_sw_sha2_256_finish` (`atcac_sha2_256_ctx` \*ctx, uint8\_t digest[**ATCA\_SHA2\_256\_DIGEST\_SIZE**])  
*Complete the SHA256 hash in software and return the digest.*
- **ATCA\_STATUS** `atcac_aes_cmac_init` (`atcac_aes_cmac_ctx` \*ctx, const uint8\_t \*key, const uint8\_t key\_len)  
*Initialize context for performing CMAC in software.*
- **ATCA\_STATUS** `atcac_aes_cmac_update` (`atcac_aes_cmac_ctx` \*ctx, const uint8\_t \*data, const size\_t data\_size)  
*Update CMAC context with input data.*
- **ATCA\_STATUS** `atcac_aes_cmac_finish` (`atcac_aes_cmac_ctx` \*ctx, uint8\_t \*cmac, size\_t \*cmac\_size)  
*Finish CMAC calculation and clear the CMAC context.*
- **ATCA\_STATUS** `atcac_sha256_hmac_init` (`atcac_hmac_sha256_ctx` \*ctx, const uint8\_t \*key, const uint8\_t key\_len)  
*Initialize context for performing HMAC (sha256) in software.*
- **ATCA\_STATUS** `atcac_sha256_hmac_update` (`atcac_hmac_sha256_ctx` \*ctx, const uint8\_t \*data, size\_t data\_size)



*Update HMAC context with input data.*

- `ATCA_STATUS atcac_sha256_hmac_finish (atcac_hmac_sha256_ctx *ctx, uint8_t *digest, size_t *digest←_len)`

*Finish CMAC calculation and clear the HMAC context.*

- `int atca_mbedtls_pk_init (mbedtls_pk_context *pkey, const uint16_t slotid)`

*Initializes an mbedtls pk context for use with EC operations.*

- `int atca_mbedtls_cert_add (mbedtls_x509_crt *cert, const atcacert_def_t *cert_def)`

*Rebuild a certificate from an atcacert\_def\_t structure, and then add it to an mbedtls cert chain.*

## 20.41.1 Detailed Description

Wrapper functions to replace cryptoauthlib software crypto functions with the mbedtls equivalent.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.41.2 Macro Definition Documentation

### 20.41.2.1 mbedtls\_calloc

```
#define mbedtls_calloc calloc
```

### 20.41.2.2 mbedtls\_free

```
#define mbedtls_free free
```

## 20.41.3 Function Documentation

### 20.41.3.1 atca\_mbedtls\_cert\_add()

```
int atca_mbedtls_cert_add (
    mbedtls_x509_crt * cert,
    const atcacert_def_t * cert_def )
```

Rebuild a certificate from an atcacert\_def\_t structure, and then add it to an mbedtls cert chain.

### Parameters

in, out	<i>cert</i>	mbedtls cert chain. Must have already been initialized
in	<i>cert_def</i>	Certificate definition that will be rebuilt and added

### Returns

0 on success, otherwise an error code.

### 20.41.3.2 atcac\_aes\_cmac\_finish()

```
ATCA_STATUS atcac_aes_cmac_finish (
    atcac_aes_cmac_ctx * ctx,
    uint8_t * cmac,
    size_t * cmac_size )
```

Finish CMAC calculation and clear the CMAC context.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### Parameters

in	<i>ctx</i>	[in] pointer to a aes-cmac context
out	<i>cmac</i>	[out] cmac value
in, out	<i>cmac_size</i>	[inout] length of cmac

### 20.41.3.3 atcac\_aes\_cmac\_init()

```
ATCA_STATUS atcac_aes_cmac_init (
    atcac_aes_cmac_ctx * ctx,
    const uint8_t * key,
    const uint8_t key_len )
```

Initialize context for performing CMAC in software.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### Parameters

in	<i>ctx</i>	[in] pointer to a aes-cmac context
in	<i>key</i>	[in] key value to use
in	<i>key_len</i>	[in] length of the key

#### 20.41.3.4 atcac\_aes\_cmac\_update()

```
ATCA_STATUS atcac_aes_cmac_update (
    atcac_aes_cmac_ctx * ctx,
    const uint8_t * data,
    const size_t data_size )
```

Update CMAC context with input data.

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

##### Parameters

in	<i>ctx</i>	[in] pointer to a aes-cmac context
in	<i>data</i>	[in] input data
in	<i>data_size</i>	[in] length of input data

#### 20.41.3.5 atcac\_aes\_gcm\_aad\_update()

```
ATCA_STATUS atcac_aes_gcm_aad_update (
    atcac_aes_gcm_ctx * ctx,
    const uint8_t * aad,
    const size_t aad_len )
```

Update the GCM context with additional authentication data (AAD)

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

##### Parameters

in	<i>ctx</i>	[in] AES-GCM Context
in	<i>aad</i>	[in] Additional Authentication Data
in	<i>aad_len</i>	[in] Length of AAD

#### 20.41.3.6 atcac\_aes\_gcm\_decrypt\_finish()

```
ATCA_STATUS atcac_aes_gcm_decrypt_finish (
    atcac_aes_gcm_ctx * ctx,
```

```
const uint8_t * tag,
size_t tag_len,
bool * is_verified )
```

Compare the AES-GCM tag and free the context.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### Parameters

in	<i>ctx</i>	[in] AES-GCM Context
in	<i>tag</i>	[in] GCM Tag to Verify
in	<i>tag_len</i>	[in] Length of the GCM tag
out	<i>is_verified</i>	[out] Tag verified as matching

### 20.41.3.7 atcac\_aes\_gcm\_decrypt\_start()

```
ATCA_STATUS atcac_aes_gcm_decrypt_start (
    atcac_aes_gcm_ctx * ctx,
    const uint8_t * key,
    const uint8_t key_len,
    const uint8_t * iv,
    const uint8_t iv_len )
```

Initialize an AES-GCM context for decryption.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### Parameters

in	<i>ctx</i>	[in] AES-GCM Context
in	<i>key</i>	[in] AES Key
in	<i>key_len</i>	[in] Length of the AES key - should be 16 or 32
in	<i>iv</i>	[in] Initialization vector input
in	<i>iv_len</i>	[in] Length of the initialization vector

### 20.41.3.8 atcac\_aes\_gcm\_decrypt\_update()

```
ATCA_STATUS atcac_aes_gcm_decrypt_update (
    atcac_aes_gcm_ctx * ctx,
```

```

const uint8_t * ciphertext,
const size_t ct_len,
uint8_t * plaintext,
size_t * pt_len )

```

Decrypt ciphertext using the initialized context.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### Parameters

in	<i>ctx</i>	[in] AES-GCM Context
in	<i>ciphertext</i>	[in] Ciphertext to decrypt
in	<i>ct_len</i>	[in] Length of the ciphertext
out	<i>plaintext</i>	[out] Resulting decrypted plaintext
in, out	<i>pt_len</i>	[inout] Length of the plaintext buffer

### 20.41.3.9 atcac\_aes\_gcm\_encrypt\_finish()

```

ATCA_STATUS atcac_aes_gcm_encrypt_finish (
    atcac_aes_gcm_ctx * ctx,
    uint8_t * tag,
    size_t tag_len )

```

Get the AES-GCM tag and free the context.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### Parameters

in	<i>ctx</i>	[in] AES-GCM Context
out	<i>tag</i>	[out] GCM Tag Result
in	<i>tag_len</i>	[in] Length of the GCM tag

### 20.41.3.10 atcac\_aes\_gcm\_encrypt\_start()

```

ATCA_STATUS atcac_aes_gcm_encrypt_start (
    atcac_aes_gcm_ctx * ctx,
    const uint8_t * key,
    const uint8_t key_len,

```

## 20.41 atca\_mbedtls\_wrap.c File Reference

---

```
const uint8_t * iv,  
const uint8_t iv_len )
```

Initialize an AES-GCM context.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### Parameters

in	<i>ctx</i>	[in] AES-GCM Context
in	<i>key</i>	[in] AES Key
in	<i>key_len</i>	[in] Length of the AES key - should be 16 or 32
in	<i>iv</i>	[in] Initialization vector input
in	<i>iv_len</i>	[in] Length of the initialization vector

### 20.41.3.11 atcac\_aes\_gcm\_encrypt\_update()

```
ATCA_STATUS atcac_aes_gcm_encrypt_update (  
    atcac_aes_gcm_ctx * ctx,  
    const uint8_t * plaintext,  
    const size_t pt_len,  
    uint8_t * ciphertext,  
    size_t * ct_len )
```

Encrypt a data using the initialized context.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### Parameters

in	<i>ctx</i>	[in] AES-GCM Context
in	<i>plaintext</i>	[in] Input buffer to encrypt
in	<i>pt_len</i>	[in] Length of the input
out	<i>ciphertext</i>	[out] Output buffer
in, out	<i>ct_len</i>	[inout] Length of the ciphertext buffer

### 20.41.3.12 atcac\_sw\_sha1\_finish()

```
int atcac_sw_sha1_finish (  
    atcac_sha1_ctx * ctx,  
    uint8_t digest[ATCA_SHA1_DIGEST_SIZE] )
```

Complete the SHA1 hash in software and return the digest.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**Parameters**

in	<i>ctx</i>	[in] pointer to a hash context
out	<i>digest</i>	[out] output buffer (20 bytes)

**20.41.3.13 atcac\_sw\_sha2\_256\_finish()**

```
int atcac_sw_sha2_256_finish (
    atcac_sha2_256_ctx * ctx,
    uint8_t digest[ATCA_SHA2_256_DIGEST_SIZE] )
```

Complete the SHA256 hash in software and return the digest.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**Parameters**

in	<i>ctx</i>	[in] pointer to a hash context
out	<i>digest</i>	[out] output buffer (32 bytes)

**20.42 atca\_mbedtls\_wrap.h File Reference****Functions**

- int [atca\\_mbedtls\\_pk\\_init](#) (struct mbedtls\_pk\_context \*pkey, const uint16\_t slotid)  
*Initializes an mbedtls pk context for use with EC operations.*
- int [atca\\_mbedtls\\_cert\\_add](#) (struct mbedtls\_x509\_crt \*cert, const struct [atcacert\\_def\\_s](#) \*cert\_def)
- int [atca\\_mbedtls\\_ecdh\\_slot\\_cb](#) (void)  
*ECDH Callback to obtain the "slot" used in ECDH operations from the application.*
- int [atca\\_mbedtls\\_ecdh\\_ioprot\\_cb](#) (uint8\_t secret[32])  
*ECDH Callback to obtain the IO Protection secret from the application.*

**20.43 atca\_openssl\_interface.c File Reference**

Crypto abstraction functions for external host side cryptography.

```
#include "atca_config.h"
#include "atca_status.h"
```

```
#include "crypto/atca_crypto_sw.h"
#include <openssl/cmac.h>
#include <openssl/evp.h>
#include <openssl/hmac.h>
```

## Functions

- [ATCA\\_STATUS atcac\\_aes\\_gcm\\_aad\\_update](#) ([atcac\\_aes\\_gcm\\_ctx](#) \*ctx, const uint8\_t \*aad, const size\_t aad\_len)  
*Update the GCM context with additional authentication data (AAD)*
- [ATCA\\_STATUS atcac\\_aes\\_gcm\\_encrypt\\_start](#) ([atcac\\_aes\\_gcm\\_ctx](#) \*ctx, const uint8\_t \*key, const uint8\_t key\_len, const uint8\_t \*iv, const uint8\_t iv\_len)  
*Initialize an AES-GCM context.*
- [ATCA\\_STATUS atcac\\_aes\\_gcm\\_encrypt\\_update](#) ([atcac\\_aes\\_gcm\\_ctx](#) \*ctx, const uint8\_t \*plaintext, const size\_t pt\_len, uint8\_t \*ciphertext, size\_t \*ct\_len)  
*Encrypt a data using the initialized context.*
- [ATCA\\_STATUS atcac\\_aes\\_gcm\\_encrypt\\_finish](#) ([atcac\\_aes\\_gcm\\_ctx](#) \*ctx, uint8\_t \*tag, size\_t tag\_len)  
*Get the AES-GCM tag and free the context.*
- [ATCA\\_STATUS atcac\\_aes\\_gcm\\_decrypt\\_start](#) ([atcac\\_aes\\_gcm\\_ctx](#) \*ctx, const uint8\_t \*key, const uint8\_t key\_len, const uint8\_t \*iv, const uint8\_t iv\_len)  
*Initialize an AES-GCM context for decryption.*
- [ATCA\\_STATUS atcac\\_aes\\_gcm\\_decrypt\\_update](#) ([atcac\\_aes\\_gcm\\_ctx](#) \*ctx, const uint8\_t \*ciphertext, const size\_t ct\_len, uint8\_t \*plaintext, size\_t \*pt\_len)  
*Decrypt ciphertext using the initialized context.*
- [ATCA\\_STATUS atcac\\_aes\\_gcm\\_decrypt\\_finish](#) ([atcac\\_aes\\_gcm\\_ctx](#) \*ctx, const uint8\_t \*tag, size\_t tag\_len, bool \*is\_verified)  
*Compare the AES-GCM tag and free the context.*
- int [atcac\\_sw\\_sha1\\_init](#) ([atcac\\_sha1\\_ctx](#) \*ctx)  
*Initialize context for performing SHA1 hash in software.*
- int [atcac\\_sw\\_sha1\\_update](#) ([atcac\\_sha1\\_ctx](#) \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Add data to a SHA1 hash.*
- int [atcac\\_sw\\_sha1\\_finish](#) ([atcac\\_sha1\\_ctx](#) \*ctx, uint8\_t digest[ATCA\_SHA1\_DIGEST\_SIZE])  
*Complete the SHA1 hash in software and return the digest.*
- int [atcac\\_sw\\_sha2\\_256\\_init](#) ([atcac\\_sha2\\_256\\_ctx](#) \*ctx)  
*Initialize context for performing SHA256 hash in software.*
- int [atcac\\_sw\\_sha2\\_256\\_update](#) ([atcac\\_sha2\\_256\\_ctx](#) \*ctx, const uint8\_t \*data, size\_t data\_size)  
*Add data to a SHA256 hash.*
- int [atcac\\_sw\\_sha2\\_256\\_finish](#) ([atcac\\_sha2\\_256\\_ctx](#) \*ctx, uint8\_t digest[ATCA\_SHA2\_256\_DIGEST\_SIZE])  
*Complete the SHA256 hash in software and return the digest.*
- [ATCA\\_STATUS atcac\\_aes\\_cmac\\_init](#) ([atcac\\_aes\\_cmac\\_ctx](#) \*ctx, const uint8\_t \*key, const uint8\_t key\_len)  
*Initialize context for performing CMAC in software.*
- [ATCA\\_STATUS atcac\\_aes\\_cmac\\_update](#) ([atcac\\_aes\\_cmac\\_ctx](#) \*ctx, const uint8\_t \*data, const size\_t data\_size)  
*Update CMAC context with input data.*
- [ATCA\\_STATUS atcac\\_aes\\_cmac\\_finish](#) ([atcac\\_aes\\_cmac\\_ctx](#) \*ctx, uint8\_t \*cmac, size\_t \*cmac\_size)  
*Finish CMAC calculation and clear the CMAC context.*
- [ATCA\\_STATUS atcac\\_sha256\\_hmac\\_init](#) ([atcac\\_hmac\\_sha256\\_ctx](#) \*ctx, const uint8\_t \*key, const uint8\_t key\_len)  
*Initialize context for performing HMAC (sha256) in software.*
- [ATCA\\_STATUS atcac\\_sha256\\_hmac\\_update](#) ([atcac\\_hmac\\_sha256\\_ctx](#) \*ctx, const uint8\_t \*data, size\_t data\_size)



*Update HMAC context with input data.*

- `ATCA_STATUS atcac_sha256_hmac_finish (atcac_hmac_sha256_ctx *ctx, uint8_t *digest, size_t *digest←_len)`

*Finish CMAC calculation and clear the HMAC context.*

## 20.43.1 Detailed Description

Crypto abstraction functions for external host side cryptography.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.43.2 Function Documentation

### 20.43.2.1 atcac\_aes\_cmac\_finish()

```
ATCA_STATUS atcac_aes_cmac_finish (
    atcac_aes_cmac_ctx * ctx,
    uint8_t * cmac,
    size_t * cmac_size )
```

Finish CMAC calculation and clear the CMAC context.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### Parameters

in	ctx	[in] pointer to a aes-cmac context
out	cmac	[out] cmac value
in, out	cmac_size	[inout] length of cmac

### 20.43.2.2 atcac\_aes\_cmac\_init()

```
ATCA_STATUS atcac_aes_cmac_init (
    atcac_aes_cmac_ctx * ctx,
    const uint8_t * key,
    const uint8_t key_len )
```

Initialize context for performing CMAC in software.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### Parameters

in	<i>ctx</i>	[in] pointer to a aes-cmac context
in	<i>key</i>	[in] key value to use
in	<i>key_len</i>	[in] length of the key

### 20.43.2.3 atcac\_aes\_cmac\_update()

```
ATCA_STATUS atcac_aes_cmac_update (
    atcac_aes_cmac_ctx * ctx,
    const uint8_t * data,
    const size_t data_size )
```

Update CMAC context with input data.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### Parameters

in	<i>ctx</i>	[in] pointer to a aes-cmac context
in	<i>data</i>	[in] input data
in	<i>data_size</i>	[in] length of input data

### 20.43.2.4 atcac\_aes\_gcm\_aad\_update()

```
ATCA_STATUS atcac_aes_gcm_aad_update (
    atcac_aes_gcm_ctx * ctx,
    const uint8_t * aad,
    const size_t aad_len )
```

Update the GCM context with additional authentication data (AAD)

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### Parameters

in	<i>ctx</i>	[in] AES-GCM Context
in	<i>aad</i>	[in] Additional Authentication Data
in	<i>aad_len</i>	[in] Length of AAD

### 20.43.2.5 atcac\_aes\_gcm\_decrypt\_finish()

```
ATCA_STATUS atcac_aes_gcm_decrypt_finish (
    atcac_aes_gcm_ctx * ctx,
    const uint8_t * tag,
    size_t tag_len,
    bool * is_verified )
```

Compare the AES-GCM tag and free the context.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### Parameters

in	<i>ctx</i>	[in] AES-GCM Context
in	<i>tag</i>	[in] GCM Tag to Verify
in	<i>tag_len</i>	[in] Length of the GCM tag
out	<i>is_verified</i>	[out] Tag verified as matching

### 20.43.2.6 atcac\_aes\_gcm\_decrypt\_start()

```
ATCA_STATUS atcac_aes_gcm_decrypt_start (
    atcac_aes_gcm_ctx * ctx,
    const uint8_t * key,
    const uint8_t key_len,
    const uint8_t * iv,
    const uint8_t iv_len )
```

Initialize an AES-GCM context for decryption.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### Parameters

in	<i>ctx</i>	[in] AES-GCM Context
in	<i>key</i>	[in] AES Key
in	<i>key_len</i>	[in] Length of the AES key - should be 16 or 32
in	<i>iv</i>	[in] Initialization vector input
in	<i>iv_len</i>	[in] Length of the initialization vector

### 20.43.2.7 atcac\_aes\_gcm\_decrypt\_update()

```
ATCA_STATUS atcac_aes_gcm_decrypt_update (
    atcac_aes_gcm_ctx * ctx,
    const uint8_t * ciphertext,
    const size_t ct_len,
    uint8_t * plaintext,
    size_t * pt_len )
```

Decrypt ciphertext using the initialized context.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### Parameters

in	<i>ctx</i>	[in] AES-GCM Context
in	<i>ciphertext</i>	[in] Ciphertext to decrypt
in	<i>ct_len</i>	[in] Length of the ciphertext
out	<i>plaintext</i>	[out] Resulting decrypted plaintext
in, out	<i>pt_len</i>	[inout] Length of the plaintext buffer

### 20.43.2.8 atcac\_aes\_gcm\_encrypt\_finish()

```
ATCA_STATUS atcac_aes_gcm_encrypt_finish (
    atcac_aes_gcm_ctx * ctx,
    uint8_t * tag,
    size_t tag_len )
```

Get the AES-GCM tag and free the context.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### Parameters

in	<i>ctx</i>	[in] AES-GCM Context
out	<i>tag</i>	[out] GCM Tag Result
in	<i>tag_len</i>	[in] Length of the GCM tag

### 20.43.2.9 atcac\_aes\_gcm\_encrypt\_start()

```
ATCA_STATUS atcac_aes_gcm_encrypt_start (
    atcac_aes_gcm_ctx * ctx,
```

```

const uint8_t * key,
const uint8_t key_len,
const uint8_t * iv,
const uint8_t iv_len )

```

Initialize an AES-GCM context.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### Parameters

in	<i>ctx</i>	[in] AES-GCM Context
in	<i>key</i>	[in] AES Key
in	<i>key_len</i>	[in] Length of the AES key - should be 16 or 32
in	<i>iv</i>	[in] Initialization vector input
in	<i>iv_len</i>	[in] Length of the initialization vector

#### 20.43.2.10 atcac\_aes\_gcm\_encrypt\_update()

```

ATCA_STATUS atcac_aes_gcm_encrypt_update (
    atcac_aes_gcm_ctx * ctx,
    const uint8_t * plaintext,
    const size_t pt_len,
    uint8_t * ciphertext,
    size_t * ct_len )

```

Encrypt a data using the initialized context.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### Parameters

in	<i>ctx</i>	[in] AES-GCM Context
in	<i>plaintext</i>	[in] Input buffer to encrypt
in	<i>pt_len</i>	[in] Length of the input
out	<i>ciphertext</i>	[out] Output buffer
in, out	<i>ct_len</i>	[inout] Length of the ciphertext buffer

#### 20.43.2.11 atcac\_sw\_sha1\_finish()

```

int atcac_sw_sha1_finish (

```

## 20.44 atca\_start\_config.h File Reference

---

```
atcac_shal_ctx * ctx,  
uint8_t digest[ATCA_SHA1_DIGEST_SIZE] )
```

Complete the SHA1 hash in software and return the digest.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### Parameters

in	ctx	[in] pointer to a hash context
out	digest	[out] output buffer (20 bytes)

## 20.43.2.12 atcac\_sw\_sha2\_256\_finish()

```
int atcac_sw_sha2_256_finish (  
    atcac_sha2_256_ctx * ctx,  
    uint8_t digest[ATCA_SHA2_256_DIGEST_SIZE] )
```

Complete the SHA256 hash in software and return the digest.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### Parameters

in	ctx	[in] pointer to a hash context
out	digest	[out] output buffer (32 bytes)

## 20.44 atca\_start\_config.h File Reference

## 20.45 atca\_start\_iface.h File Reference

## 20.46 atca\_status.h File Reference

Microchip Crypto Auth status codes.

```
#include <stdint.h>  
#include "atca_bool.h"
```

## Macros

- `#define ATCA_STATUS_AUTH_BIT 0x40`

## Enumerations

- `enum ATCA_STATUS {`  
`ATCA_SUCCESS = 0x00, ATCA_CONFIG_ZONE_LOCKED = 0x01, ATCA_DATA_ZONE_LOCKED =`  
`0x02, ATCA_INVALID_POINTER,`  
`ATCA_INVALID_LENGTH, ATCA_WAKE_FAILED = 0xD0, ATCA_CHECKMAC_VERIFY_FAILED = 0xD1,`  
`ATCA_PARSE_ERROR = 0xD2,`  
`ATCA_STATUS_CRC = 0xD4, ATCA_STATUS_UNKNOWN = 0xD5, ATCA_STATUS_ECC = 0xD6,`  
`ATCA_STATUS_SELFTEST_ERROR = 0xD7,`  
`ATCA_FUNC_FAIL = 0xE0, ATCA_GEN_FAIL = 0xE1, ATCA_BAD_PARAM = 0xE2, ATCA_INVALID_ID =`  
`0xE3,`  
`ATCA_INVALID_SIZE = 0xE4, ATCA_RX_CRC_ERROR = 0xE5, ATCA_RX_FAIL = 0xE6, ATCA_RX_NO_RESPONSE`  
`= 0xE7,`  
`ATCA_RESYNC_WITH_WAKEUP = 0xE8, ATCA_PARITY_ERROR = 0xE9, ATCA_TX_TIMEOUT = 0xEA,`  
`ATCA_RX_TIMEOUT = 0xEB,`  
`ATCA_TOO_MANY_COMM_RETRIES = 0xEC, ATCA_SMALL_BUFFER = 0xED, ATCA_COMM_FAIL =`  
`0xF0, ATCA_TIMEOUT = 0xF1,`  
`ATCA_BAD_OPCODE = 0xF2, ATCA_WAKE_SUCCESS = 0xF3, ATCA_EXECUTION_ERROR = 0xF4,`  
`ATCA_UNIMPLEMENTED = 0xF5,`  
`ATCA_ASSERT_FAILURE = 0xF6, ATCA_TX_FAIL = 0xF7, ATCA_NOT_LOCKED = 0xF8, ATCA_NO_DEVICES`  
`= 0xF9,`  
`ATCA_HEALTH_TEST_ERROR = 0xFA, ATCA_ALLOC_FAILURE = 0xFB, ATCA_USE_FLAGS_CONSUMED`  
`= 0xFC, ATCA_NOT_INITIALIZED = 0xFD }`

### 20.46.1 Detailed Description

Microchip Crypto Auth status codes.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.46.2 Macro Definition Documentation

#### 20.46.2.1 ATCA\_STATUS\_AUTH\_BIT

```
#define ATCA_STATUS_AUTH_BIT 0x40
```

### 20.46.3 Enumeration Type Documentation

#### 20.46.3.1 ATCA\_STATUS

```
enum ATCA_STATUS
```

## Enumerator

ATCA_SUCCESS	Function succeeded.
ATCA_CONFIG_ZONE_LOCKED	
ATCA_DATA_ZONE_LOCKED	
ATCA_INVALID_POINTER	
ATCA_INVALID_LENGTH	
ATCA_WAKE_FAILED	response status byte indicates CheckMac failure (status byte = 0x01)
ATCA_CHECKMAC_VERIFY_FAILED	response status byte indicates CheckMac failure (status byte = 0x01)
ATCA_PARSE_ERROR	response status byte indicates parsing error (status byte = 0x03)
ATCA_STATUS_CRC	response status byte indicates DEVICE did not receive data properly (status byte = 0xFF)
ATCA_STATUS_UNKNOWN	response status byte is unknown
ATCA_STATUS_ECC	response status byte is ECC fault (status byte = 0x05)
ATCA_STATUS_SELFTEST_ERROR	response status byte is Self Test Error, chip in failure mode (status byte = 0x07)
ATCA_FUNC_FAIL	Function could not execute due to incorrect condition / state.
ATCA_GEN_FAIL	unspecified error
ATCA_BAD_PARAM	bad argument (out of range, null pointer, etc.)
ATCA_INVALID_ID	invalid device id, id not set
ATCA_INVALID_SIZE	Count value is out of range or greater than buffer size.
ATCA_RX_CRC_ERROR	CRC error in data received from device.
ATCA_RX_FAIL	Timed out while waiting for response. Number of bytes received is > 0.
ATCA_RX_NO_RESPONSE	Not an error while the Command layer is polling for a command response.
ATCA_RESYNC_WITH_WAKEUP	Re-synchronization succeeded, but only after generating a Wake-up.
ATCA_PARITY_ERROR	for protocols needing parity
ATCA_TX_TIMEOUT	for Microchip PHY protocol, timeout on transmission waiting for master
ATCA_RX_TIMEOUT	for Microchip PHY protocol, timeout on receipt waiting for master
ATCA_TOO_MANY_COMM_RETRIES	Device did not respond too many times during a transmission. Could indicate no device present.
ATCA_SMALL_BUFFER	Supplied buffer is too small for data required.
ATCA_COMM_FAIL	Communication with device failed. Same as in hardware dependent modules.
ATCA_TIMEOUT	Timed out while waiting for response. Number of bytes received is 0.
ATCA_BAD_OPCODE	opcode is not supported by the device
ATCA_WAKE_SUCCESS	received proper wake token
ATCA_EXECUTION_ERROR	chip was in a state where it could not execute the command, response status byte indicates command execution error (status byte = 0x0F)
ATCA_UNIMPLEMENTED	Function or some element of it hasn't been implemented yet.
ATCA_ASSERT_FAILURE	Code failed run-time consistency check.
ATCA_TX_FAIL	Failed to write.
ATCA_NOT_LOCKED	required zone was not locked
ATCA_NO_DEVICES	For protocols that support device discovery (kit protocol), no devices were found.
ATCA_HEALTH_TEST_ERROR	random number generator health test error



**Enumerator**

ATCA_ALLOC_FAILURE	Couldn't allocate required memory.
ATCA_USE_FLAGS_CONSUMED	Use flags on the device indicates its consumed fully.
ATCA_NOT_INITIALIZED	The library has not been initialized so the command could not be executed.

## 20.47 atca\_version.h File Reference

Microchip CryptoAuth Library Version.

### Macros

- #define ATCA\_LIBRARY\_VERSION\_DATE "20200728"
- #define ATCA\_LIBRARY\_VERSION\_MAJOR 3
- #define ATCA\_LIBRARY\_VERSION\_MINOR 2
- #define ATCA\_LIBRARY\_VERSION\_BUILD 2

### 20.47.1 Detailed Description

Microchip CryptoAuth Library Version.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.47.2 Macro Definition Documentation

#### 20.47.2.1 ATCA\_LIBRARY\_VERSION\_BUILD

```
#define ATCA_LIBRARY_VERSION_BUILD 2
```

#### 20.47.2.2 ATCA\_LIBRARY\_VERSION\_DATE

```
#define ATCA_LIBRARY_VERSION_DATE "20200728"
```

## 20.48 atca\_wolfssl\_interface.c File Reference

---

### 20.47.2.3 ATCA\_LIBRARY\_VERSION\_MAJOR

```
#define ATCA_LIBRARY_VERSION_MAJOR 3
```

### 20.47.2.4 ATCA\_LIBRARY\_VERSION\_MINOR

```
#define ATCA_LIBRARY_VERSION_MINOR 2
```

## 20.48 atca\_wolfssl\_interface.c File Reference

Crypto abstraction functions for external host side cryptography.

```
#include "atca_config.h"  
#include "atca_status.h"  
#include "crypto/atca_crypto_sw.h"
```

### 20.48.1 Detailed Description

Crypto abstraction functions for external host side cryptography.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.49 atcacert.h File Reference

Declarations common to all atcacert code.

```
#include <stddef.h>  
#include <stdint.h>
```

## Macros

- #define `FALSE` (0)
- #define `TRUE` (1)
- #define `ATCACERT_E_SUCCESS` 0  
*Operation completed successfully.*
- #define `ATCACERT_E_ERROR` 1  
*General error.*
- #define `ATCACERT_E_BAD_PARAMS` 2  
*Invalid/bad parameter passed to function.*
- #define `ATCACERT_E_BUFFER_TOO_SMALL` 3  
*Supplied buffer for output is too small to hold the result.*
- #define `ATCACERT_E_DECODING_ERROR` 4  
*Data being decoded/parsed has an invalid format.*
- #define `ATCACERT_E_INVALID_DATE` 5  
*Date is invalid.*
- #define `ATCACERT_E_UNIMPLEMENTED` 6  
*Function is unimplemented for the current configuration.*
- #define `ATCACERT_E_UNEXPECTED_ELEM_SIZE` 7  
*A certificate element size was not what was expected.*
- #define `ATCACERT_E_ELEM_MISSING` 8  
*The certificate element isn't defined for the certificate definition.*
- #define `ATCACERT_E_ELEM_OUT_OF_BOUNDS` 9  
*Certificate element is out of bounds for the given certificate.*
- #define `ATCACERT_E_BAD_CERT` 10  
*Certificate structure is bad in some way.*
- #define `ATCACERT_E_WRONG_CERT_DEF` 11
- #define `ATCACERT_E_VERIFY_FAILED` 12  
*Certificate or challenge/response verification failed.*
- #define `ATCACERT_E_INVALID_TRANSFORM` 13  
*Invalid transform passed to function.*

### 20.49.1 Detailed Description

Declarations common to all atcacert code.

These are common definitions used by all the atcacert code.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.50 atcacert\_client.c File Reference

Client side cert i/o methods. These declarations deal with the client-side, the node being authenticated, of the authentication process. It is assumed the client has an ECC CryptoAuthentication device (e.g. ATECC508A) and the certificates are stored on that device.

```
#include <stdlib.h>
#include "atcacert_client.h"
#include "atcacert_der.h"
#include "atcacert_pem.h"
#include "cryptoauthlib.h"
#include "calib/calib_basic.h"
```

## Functions

- int [atcacert\\_get\\_response](#) (uint8\_t device\_private\_key\_slot, const uint8\_t challenge[32], uint8\_t response[64])  
*Calculates the response to a challenge sent from the host.*
- int [atcacert\\_read\\_device\\_loc](#) (const [atcacert\\_device\\_loc\\_t](#) \*device\_loc, uint8\_t \*data)  
*Read the data from a device location.*
- int [atcacert\\_read\\_cert](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t ca\_public\_key[64], uint8\_t \*cert, size\_t \*cert\_size)  
*Reads the certificate specified by the certificate definition from the ATECC508A device.*
- int [atcacert\\_write\\_cert](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size)  
*Take a full certificate and write it to the ATECC508A device according to the certificate definition.*
- int [atcacert\\_create\\_csr\\_pem](#) (const [atcacert\\_def\\_t](#) \*csr\_def, char \*csr, size\_t \*csr\_size)  
*Creates a CSR specified by the CSR definition from the ATECC508A device. This process involves reading the dynamic CSR data from the device and combining it with the template found in the CSR definition, then signing it. Return the CSR in der format.*
- int [atcacert\\_create\\_csr](#) (const [atcacert\\_def\\_t](#) \*csr\_def, uint8\_t \*csr, size\_t \*csr\_size)  
*Creates a CSR specified by the CSR definition from the ATECC508A device. This process involves reading the dynamic CSR data from the device and combining it with the template found in the CSR definition, then signing it. Return the CSR in der format.*
- int [atcacert\\_read\\_subj\\_key\\_id](#) (const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t subj\_key\_id[20])  
*Reads the subject key ID based on a certificate definition.*
- int [atcacert\\_read\\_cert\\_size](#) (const [atcacert\\_def\\_t](#) \*cert\_def, size\_t \*cert\_size)  
*Return the actual certificate size in bytes for a given cert def. Certificate can be variable size, so this gives the absolute buffer size when reading the certificates.*

### 20.50.1 Detailed Description

Client side cert i/o methods. These declarations deal with the client-side, the node being authenticated, of the authentication process. It is assumed the client has an ECC CryptoAuthentication device (e.g. ATECC508A) and the certificates are stored on that device.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.51 atcacert\_client.h File Reference

Client side cert i/o methods. These declarations deal with the client-side, the node being authenticated, of the authentication process. It is assumed the client has an ECC CryptoAuthentication device (e.g. ATECC508A) and the certificates are stored on that device.

```
#include <stddef.h>
#include <stdint.h>
#include "atcacert_def.h"
```

## Functions

- int `atcacert_read_device_loc` (const `atcacert_device_loc_t` \*device\_loc, uint8\_t \*data)  
*Read the data from a device location.*
- int `atcacert_read_cert` (const `atcacert_def_t` \*cert\_def, const uint8\_t ca\_public\_key[64], uint8\_t \*cert, size\_t \*cert\_size)  
*Reads the certificate specified by the certificate definition from the ATECC508A device.*
- int `atcacert_write_cert` (const `atcacert_def_t` \*cert\_def, const uint8\_t \*cert, size\_t cert\_size)  
*Take a full certificate and write it to the ATECC508A device according to the certificate definition.*
- int `atcacert_create_csr` (const `atcacert_def_t` \*csr\_def, uint8\_t \*csr, size\_t \*csr\_size)  
*Creates a CSR specified by the CSR definition from the ATECC508A device. This process involves reading the dynamic CSR data from the device and combining it with the template found in the CSR definition, then signing it. Return the CSR in der format.*
- int `atcacert_create_csr_pem` (const `atcacert_def_t` \*csr\_def, char \*csr, size\_t \*csr\_size)  
*Creates a CSR specified by the CSR definition from the ATECC508A device. This process involves reading the dynamic CSR data from the device and combining it with the template found in the CSR definition, then signing it. Return the CSR in der format.*
- int `atcacert_get_response` (uint8\_t device\_private\_key\_slot, const uint8\_t challenge[32], uint8\_t \*response[64])  
*Calculates the response to a challenge sent from the host.*
- int `atcacert_read_subj_key_id` (const `atcacert_def_t` \*cert\_def, uint8\_t subj\_key\_id[20])  
*Reads the subject key ID based on a certificate definition.*
- int `atcacert_read_cert_size` (const `atcacert_def_t` \*cert\_def, size\_t \*cert\_size)  
*Return the actual certificate size in bytes for a given cert def. Certificate can be variable size, so this gives the absolute buffer size when reading the certificates.*

### 20.51.1 Detailed Description

Client side cert i/o methods. These declarations deal with the client-side, the node being authenticated, of the authentication process. It is assumed the client has an ECC CryptoAuthentication device (e.g. ATECC508A) and the certificates are stored on that device.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.52 atcacert\_date.c File Reference

Date handling with regard to certificates.

```
#include <string.h>
#include "atcacert_date.h"
```

## Functions

- int [atcacert\\_date\\_enc](#) ([atcacert\\_date\\_format\\_t](#) format, const [atcacert\\_tm\\_utc\\_t](#) \*timestamp, uint8\_t \*formatted\_date, size\_t \*formatted\_date\_size)  
*Format a timestamp according to the format type.*
- int [atcacert\\_date\\_dec](#) ([atcacert\\_date\\_format\\_t](#) format, const uint8\_t \*formatted\_date, size\_t formatted\_date\_size, [atcacert\\_tm\\_utc\\_t](#) \*timestamp)  
*Parse a formatted timestamp according to the specified format.*
- int [atcacert\\_date\\_get\\_max\\_date](#) ([atcacert\\_date\\_format\\_t](#) format, [atcacert\\_tm\\_utc\\_t](#) \*timestamp)  
*Return the maximum date available for the given format.*
- int [atcacert\\_date\\_enc\\_iso8601\\_sep](#) (const [atcacert\\_tm\\_utc\\_t](#) \*timestamp, uint8\_t formatted\_date[(20)])
- int [atcacert\\_date\\_dec\\_iso8601\\_sep](#) (const uint8\_t formatted\_date[(20)], [atcacert\\_tm\\_utc\\_t](#) \*timestamp)
- int [atcacert\\_date\\_enc\\_rfc5280\\_utc](#) (const [atcacert\\_tm\\_utc\\_t](#) \*timestamp, uint8\_t formatted\_date[(13)])
- int [atcacert\\_date\\_dec\\_rfc5280\\_utc](#) (const uint8\_t formatted\_date[(13)], [atcacert\\_tm\\_utc\\_t](#) \*timestamp)
- int [atcacert\\_date\\_enc\\_rfc5280\\_gen](#) (const [atcacert\\_tm\\_utc\\_t](#) \*timestamp, uint8\_t formatted\_date[(15)])
- int [atcacert\\_date\\_dec\\_rfc5280\\_gen](#) (const uint8\_t formatted\_date[(15)], [atcacert\\_tm\\_utc\\_t](#) \*timestamp)
- int [atcacert\\_date\\_enc\\_posix\\_uint32\\_be](#) (const [atcacert\\_tm\\_utc\\_t](#) \*timestamp, uint8\_t formatted\_date[(4)])
- int [atcacert\\_date\\_dec\\_posix\\_uint32\\_be](#) (const uint8\_t formatted\_date[(4)], [atcacert\\_tm\\_utc\\_t](#) \*timestamp)
- int [atcacert\\_date\\_enc\\_posix\\_uint32\\_le](#) (const [atcacert\\_tm\\_utc\\_t](#) \*timestamp, uint8\_t formatted\_date[(4)])
- int [atcacert\\_date\\_dec\\_posix\\_uint32\\_le](#) (const uint8\_t formatted\_date[(4)], [atcacert\\_tm\\_utc\\_t](#) \*timestamp)
- int [atcacert\\_date\\_enc\\_compcert](#) (const [atcacert\\_tm\\_utc\\_t](#) \*issue\_date, uint8\_t expire\_years, uint8\_t enc\_dates[3])  
*Encode the issue and expire dates in the format used by the compressed certificate.*
- int [atcacert\\_date\\_dec\\_compcert](#) (const uint8\_t enc\_dates[3], [atcacert\\_date\\_format\\_t](#) expire\_date\_format, [atcacert\\_tm\\_utc\\_t](#) \*issue\_date, [atcacert\\_tm\\_utc\\_t](#) \*expire\_date)  
*Decode the issue and expire dates from the format used by the compressed certificate.*

## Variables

- const size\_t [ATCACERT\\_DATE\\_FORMAT\\_SIZES](#) [5]

### 20.52.1 Detailed Description

Date handling with regard to certificates.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.53 atcacert\_date.h File Reference

Declarations for date handling with regard to certificates.

```
#include <stddef.h>
#include "atcacert.h"
```

## Data Structures

- struct [atcacert\\_tm\\_utc\\_s](#)

## Macros

- #define `DATEFMT_ISO8601_SEP_SIZE` (20)
- #define `DATEFMT_RFC5280_UTC_SIZE` (13)
- #define `DATEFMT_POSIX_UINT32_BE_SIZE` (4)
- #define `DATEFMT_POSIX_UINT32_LE_SIZE` (4)
- #define `DATEFMT_RFC5280_GEN_SIZE` (15)
- #define `DATEFMT_MAX_SIZE` `DATEFMT_ISO8601_SEP_SIZE`
- #define `ATCACERT_DATE_FORMAT_SIZES_COUNT` 5

## Typedefs

- typedef struct `atcacert_tm_utc_s` `atcacert_tm_utc_t`
- typedef enum `atcacert_date_format_e` `atcacert_date_format_t`

## Enumerations

- enum `atcacert_date_format_e` {  
`DATEFMT_ISO8601_SEP`, `DATEFMT_RFC5280_UTC`, `DATEFMT_POSIX_UINT32_BE`, `DATEFMT_POSIX_UINT32_LE`,  
`DATEFMT_RFC5280_GEN` }

## Functions

- int `atcacert_date_enc` (`atcacert_date_format_t` format, const `atcacert_tm_utc_t` \*timestamp, uint8\_t \*formatted\_date, size\_t \*formatted\_date\_size)  
*Format a timestamp according to the format type.*
- int `atcacert_date_dec` (`atcacert_date_format_t` format, const uint8\_t \*formatted\_date, size\_t formatted\_date\_size, `atcacert_tm_utc_t` \*timestamp)  
*Parse a formatted timestamp according to the specified format.*
- int `atcacert_date_enc_compcert` (const `atcacert_tm_utc_t` \*issue\_date, uint8\_t expire\_years, uint8\_t enc\_dates[3])  
*Encode the issue and expire dates in the format used by the compressed certificate.*
- int `atcacert_date_dec_compcert` (const uint8\_t enc\_dates[3], `atcacert_date_format_t` expire\_date\_format, `atcacert_tm_utc_t` \*issue\_date, `atcacert_tm_utc_t` \*expire\_date)  
*Decode the issue and expire dates from the format used by the compressed certificate.*
- int `atcacert_date_get_max_date` (`atcacert_date_format_t` format, `atcacert_tm_utc_t` \*timestamp)  
*Return the maximum date available for the given format.*
- int `atcacert_date_enc_iso8601_sep` (const `atcacert_tm_utc_t` \*timestamp, uint8\_t formatted\_date[(20)])
- int `atcacert_date_dec_iso8601_sep` (const uint8\_t formatted\_date[(20)], `atcacert_tm_utc_t` \*timestamp)
- int `atcacert_date_enc_rfc5280_utc` (const `atcacert_tm_utc_t` \*timestamp, uint8\_t formatted\_date[(13)])
- int `atcacert_date_dec_rfc5280_utc` (const uint8\_t formatted\_date[(13)], `atcacert_tm_utc_t` \*timestamp)
- int `atcacert_date_enc_rfc5280_gen` (const `atcacert_tm_utc_t` \*timestamp, uint8\_t formatted\_date[(15)])
- int `atcacert_date_dec_rfc5280_gen` (const uint8\_t formatted\_date[(15)], `atcacert_tm_utc_t` \*timestamp)
- int `atcacert_date_enc_posix_uint32_be` (const `atcacert_tm_utc_t` \*timestamp, uint8\_t formatted\_date[(4)])
- int `atcacert_date_dec_posix_uint32_be` (const uint8\_t formatted\_date[(4)], `atcacert_tm_utc_t` \*timestamp)
- int `atcacert_date_enc_posix_uint32_le` (const `atcacert_tm_utc_t` \*timestamp, uint8\_t formatted\_date[(4)])
- int `atcacert_date_dec_posix_uint32_le` (const uint8\_t formatted\_date[(4)], `atcacert_tm_utc_t` \*timestamp)

## Variables

- const size\_t `ATCACERT_DATE_FORMAT_SIZES` [5]

### 20.53.1 Detailed Description

Declarations for date handling with regard to certificates.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.54 atcacert\_def.c File Reference

Main certificate definition implementation.

```
#include "atcacert_def.h"
#include "crypto/atca_crypto_sw_sha1.h"
#include "crypto/atca_crypto_sw_sha2.h"
#include "atcacert_der.h"
#include "atcacert_date.h"
#include <string.h>
#include "atca_helpers.h"
```

### Macros

- #define [ATCACERT\\_MIN](#)(x, y) ((x) < (y) ? (x) : (y))
- #define [ATCACERT\\_MAX](#)(x, y) ((x) >= (y) ? (x) : (y))

### Functions

- int [atcacert\\_merge\\_device\\_loc](#) ([atcacert\\_device\\_loc\\_t](#) \*device\_locs, size\_t \*device\_locs\_count, size\_t device\_locs\_max\_count, const [atcacert\\_device\\_loc\\_t](#) \*device\_loc, size\_t block\_size)  
*Merge a new device location into a list of device locations. If the new location overlaps with an existing location, the existing one will be modified to encompass both. Otherwise the new location is appended to the end of the list.*
- int [atcacert\\_get\\_device\\_locs](#) (const [atcacert\\_def\\_t](#) \*cert\_def, [atcacert\\_device\\_loc\\_t](#) \*device\_locs, size\_t \*device\_locs\_count, size\_t device\_locs\_max\_count, size\_t block\_size)  
*Add all the device locations required to rebuild the specified certificate (cert\_def) to a device locations list.*
- int [atcacert\\_cert\\_build\\_start](#) ([atcacert\\_build\\_state\\_t](#) \*build\_state, const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t \*cert\_size, const uint8\_t ca\_public\_key[64])  
*Starts the certificate rebuilding process.*
- int [atcacert\\_cert\\_build\\_process](#) ([atcacert\\_build\\_state\\_t](#) \*build\_state, const [atcacert\\_device\\_loc\\_t](#) \*device\_loc, const uint8\_t \*device\_data)  
*Process information read from the ATECC device. If it contains information for the certificate, it will be incorporated into the certificate.*
- int [atcacert\\_cert\\_build\\_finish](#) ([atcacert\\_build\\_state\\_t](#) \*build\_state)  
*Completes any final certificate processing required after all data from the device has been incorporated.*
- int [atcacert\\_is\\_device\\_loc\\_overlap](#) (const [atcacert\\_device\\_loc\\_t](#) \*device\_loc1, const [atcacert\\_device\\_loc\\_t](#) \*device\_loc2)  
*Determines if the two device locations overlap.*
- int [atcacert\\_get\\_device\\_data](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, const [atcacert\\_device\\_loc\\_t](#) \*device\_loc, uint8\_t \*device\_data)



*Gets the dynamic data that would be saved to the specified device location. This function is primarily used to break down a full certificate into the dynamic components to be saved to a device.*

- int [atcacert\\_set\\_subj\\_public\\_key](#) (const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t cert\_size, const uint8\_t subj\_public\_key[64])

*Sets the subject public key and subject key ID in a certificate.*

- int [atcacert\\_get\\_subj\\_public\\_key](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t subj\_public\_key[64])

*Gets the subject public key from a certificate.*

- int [atcacert\\_get\\_subj\\_key\\_id](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t subj\_key\_id[20])

*Gets the subject key ID from a certificate.*

- int [atcacert\\_set\\_signature](#) (const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t \*cert\_size, size\_t max\_cert\_size, const uint8\_t signature[64])

*Sets the signature in a certificate. This may alter the size of the X.509 certificates.*

- int [atcacert\\_get\\_signature](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t signature[64])

*Gets the signature from a certificate.*

- int [atcacert\\_set\\_issue\\_date](#) (const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t cert\_size, const [atcacert\\_tm\\_utc\\_t](#) \*timestamp)

*Sets the issue date (notBefore) in a certificate. Will be formatted according to the date format specified in the certificate definition.*

- int [atcacert\\_get\\_issue\\_date](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, [atcacert\\_tm\\_utc\\_t](#) \*timestamp)

*Gets the issue date from a certificate. Will be parsed according to the date format specified in the certificate definition.*

- int [atcacert\\_set\\_expire\\_date](#) (const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t cert\_size, const [atcacert\\_tm\\_utc\\_t](#) \*timestamp)

*Sets the expire date (notAfter) in a certificate. Will be formatted according to the date format specified in the certificate definition.*

- int [atcacert\\_get\\_expire\\_date](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, [atcacert\\_tm\\_utc\\_t](#) \*timestamp)

*Gets the expire date from a certificate. Will be parsed according to the date format specified in the certificate definition.*

- int [atcacert\\_set\\_signer\\_id](#) (const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t cert\_size, const uint8\_t signer\_id[2])

*Sets the signer ID in a certificate. Will be formatted as 4 upper-case hex digits.*

- int [atcacert\\_get\\_signer\\_id](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t signer\_id[2])

*Gets the signer ID from a certificate. Will be parsed as 4 upper-case hex digits.*

- int [atcacert\\_set\\_cert\\_sn](#) (const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t \*cert\_size, size\_t max\_cert\_size, const uint8\_t \*cert\_sn, size\_t cert\_sn\_size)

*Sets the certificate serial number in a certificate.*

- int [atcacert\\_gen\\_cert\\_sn](#) (const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t cert\_size, const uint8\_t device\_sn[9])

*Sets the certificate serial number by generating it from other information in the certificate using the scheme specified by sn\_source in cert\_def. See the.*

- int [atcacert\\_get\\_cert\\_sn](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t \*cert\_sn, size\_t \*cert\_sn\_size)

*Gets the certificate serial number from a certificate.*

- int [atcacert\\_set\\_auth\\_key\\_id](#) (const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t cert\_size, const uint8\_t auth\_public\_key[64])

*Sets the authority key ID in a certificate. Note that this takes the actual public key creates a key ID from it.*

- int [atcacert\\_set\\_auth\\_key\\_id\\_raw](#) (const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t cert\_size, const uint8\_t \*auth\_key\_id)

*Sets the authority key ID in a certificate.*

- int [atcacert\\_get\\_auth\\_key\\_id](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t auth\_key\_id[20])  
*Gets the authority key ID from a certificate.*
- int [atcacert\\_set\\_comp\\_cert](#) (const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t \*cert\_size, size\_t max\_cert\_size, const uint8\_t comp\_cert[72])  
*Sets the signature, issue date, expire date, and signer ID found in the compressed certificate. This also checks fields common between the cert\_def and the compressed certificate to make sure they match.*
- int [atcacert\\_get\\_comp\\_cert](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t comp\_cert[72])  
*Generate the compressed certificate for the given certificate.*
- int [atcacert\\_get\\_tbs](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, const uint8\_t \*\*tbs, size\_t \*tbs\_size)  
*Get a pointer to the TBS data in a certificate.*
- int [atcacert\\_get\\_tbs\\_digest](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t tbs\_digest[32])  
*Get the SHA256 digest of certificate's TBS data.*
- int [atcacert\\_set\\_cert\\_element](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const [atcacert\\_cert\\_loc\\_t](#) \*cert\_loc, uint8\_t \*cert, size\_t cert\_size, const uint8\_t \*data, size\_t data\_size)  
*Sets an element in a certificate. The data\_size must match the size in cert\_loc.*
- int [atcacert\\_get\\_cert\\_element](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const [atcacert\\_cert\\_loc\\_t](#) \*cert\_loc, const uint8\_t \*cert, size\_t cert\_size, uint8\_t \*data, size\_t data\_size)  
*Gets an element from a certificate.*
- int [atcacert\\_get\\_key\\_id](#) (const uint8\_t public\_key[64], uint8\_t key\_id[20])  
*Calculates the key ID for a given public ECC P256 key.*
- void [atcacert\\_public\\_key\\_add\\_padding](#) (const uint8\_t raw\_key[64], uint8\_t padded\_key[72])  
*Takes a raw P256 ECC public key and converts it to the padded version used by ATECC devices. Input and output buffers can point to the same location to do an in-place transform.*
- void [atcacert\\_public\\_key\\_remove\\_padding](#) (const uint8\_t padded\_key[72], uint8\_t raw\_key[64])  
*Takes a padded public key used by ATECC devices and converts it to a raw P256 ECC public key. Input and output buffers can point to the same location to do an in-place transform.*
- int [atcacert\\_transform\\_data](#) ([atcacert\\_transform\\_t](#) transform, const uint8\_t \*data, size\_t data\_size, uint8\_t \*destination, size\_t \*destination\_size)  
*Apply the specified transform to the specified data.*
- int [atcacert\\_max\\_cert\\_size](#) (const [atcacert\\_def\\_t](#) \*cert\_def, size\_t \*max\_cert\_size)  
*Return the maximum possible certificate size in bytes for a given cert def. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificates.*

### 20.54.1 Detailed Description

Main certificate definition implementation.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.54.2 Macro Definition Documentation

### 20.54.2.1 ATCACERT\_MAX

```
#define ATCACERT_MAX(  
    x,  
    y ) ((x) >= (y) ? (x) : (y))
```

### 20.54.2.2 ATCACERT\_MIN

```
#define ATCACERT_MIN(  
    x,  
    y ) ((x) < (y) ? (x) : (y))
```

## 20.55 atcacert\_def.h File Reference

Declarations for certificates related to ECC CryptoAuthentication devices. These are the definitions required to define a certificate and its various elements with regards to the CryptoAuthentication ECC devices.

```
#include <stddef.h>  
#include <stdint.h>  
#include "atca_compiler.h"  
#include "atcacert.h"  
#include "atcacert_date.h"  
#include "atca_helpers.h"
```

### Data Structures

- struct [atcacert\\_device\\_loc\\_s](#)
- struct [atcacert\\_cert\\_loc\\_s](#)
- struct [atcacert\\_cert\\_element\\_s](#)
- struct [atcacert\\_def\\_s](#)
- struct [atcacert\\_build\\_state\\_s](#)

### Macros

- #define [ATCA\\_MAX\\_TRANSFORMS](#) 2
- #define [ATCA\\_PACKED](#)

### Typedefs

- typedef enum [atcacert\\_cert\\_type\\_e](#) [atcacert\\_cert\\_type\\_t](#)
- typedef enum [atcacert\\_cert\\_sn\\_src\\_e](#) [atcacert\\_cert\\_sn\\_src\\_t](#)
- typedef enum [atcacert\\_device\\_zone\\_e](#) [atcacert\\_device\\_zone\\_t](#)
- typedef enum [atcacert\\_transform\\_e](#) [atcacert\\_transform\\_t](#)  
*How to transform the data from the device to the certificate.*
- typedef enum [atcacert\\_std\\_cert\\_element\\_e](#) [atcacert\\_std\\_cert\\_element\\_t](#)
- typedef struct [atcacert\\_device\\_loc\\_s](#) [atcacert\\_device\\_loc\\_t](#)
- typedef struct [atcacert\\_cert\\_loc\\_s](#) [atcacert\\_cert\\_loc\\_t](#)
- typedef struct [atcacert\\_cert\\_element\\_s](#) [atcacert\\_cert\\_element\\_t](#)
- typedef struct [atcacert\\_def\\_s](#) [atcacert\\_def\\_t](#)
- typedef struct [atcacert\\_build\\_state\\_s](#) [atcacert\\_build\\_state\\_t](#)

## Enumerations

- enum [atcacert\\_cert\\_type\\_e](#) { CERTTYPE\_X509, CERTTYPE\_CUSTOM }
  - enum [atcacert\\_cert\\_sn\\_src\\_e](#) {  
SNSRC\_STORED = 0x0, SNSRC\_STORED\_DYNAMIC = 0x7, SNSRC\_DEVICE\_SN = 0x8, SNSRC\_SIGNER\_ID = 0x9,  
SNSRC\_PUB\_KEY\_HASH = 0xA, SNSRC\_DEVICE\_SN\_HASH = 0xB, SNSRC\_PUB\_KEY\_HASH\_POS = 0xC, SNSRC\_DEVICE\_SN\_HASH\_POS = 0xD,  
SNSRC\_PUB\_KEY\_HASH\_RAW = 0xE, SNSRC\_DEVICE\_SN\_HASH\_RAW = 0xF }
  - enum [atcacert\\_device\\_zone\\_e](#) { DEVZONE\_CONFIG = 0x00, DEVZONE\_OTP = 0x01, DEVZONE\_DATA = 0x02, DEVZONE\_NONE = 0x07 }
  - enum [atcacert\\_transform\\_e](#) {  
TF\_NONE, TF\_REVERSE, TF\_BIN2HEX\_UC, TF\_BIN2HEX\_LC,  
TF\_HEX2BIN\_UC, TF\_HEX2BIN\_LC, TF\_BIN2HEX\_SPACE\_UC, TF\_BIN2HEX\_SPACE\_LC,  
TF\_HEX2BIN\_SPACE\_UC, TF\_HEX2BIN\_SPACE\_LC }
- How to transform the data from the device to the certificate.*
- enum [atcacert\\_std\\_cert\\_element\\_e](#) {  
STDCERT\_PUBLIC\_KEY, STDCERT\_SIGNATURE, STDCERT\_ISSUE\_DATE, STDCERT\_EXPIRE\_DATE,  
STDCERT\_SIGNER\_ID, STDCERT\_CERT\_SN, STDCERT\_AUTH\_KEY\_ID, STDCERT\_SUBJ\_KEY\_ID,  
STDCERT\_NUM\_ELEMENTS }

## Functions

- int [atcacert\\_get\\_device\\_locs](#) (const [atcacert\\_def\\_t](#) \*cert\_def, [atcacert\\_device\\_loc\\_t](#) \*device\_locs, size\_t \*device\_locs\_count, size\_t device\_locs\_max\_count, size\_t block\_size)  
*Add all the device locations required to rebuild the specified certificate (cert\_def) to a device locations list.*
- int [atcacert\\_cert\\_build\\_start](#) ([atcacert\\_build\\_state\\_t](#) \*build\_state, const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t \*cert\_size, const uint8\_t ca\_public\_key[64])  
*Starts the certificate rebuilding process.*
- int [atcacert\\_cert\\_build\\_process](#) ([atcacert\\_build\\_state\\_t](#) \*build\_state, const [atcacert\\_device\\_loc\\_t](#) \*device\_loc, const uint8\_t \*device\_data)  
*Process information read from the ATECC device. If it contains information for the certificate, it will be incorporated into the certificate.*
- int [atcacert\\_cert\\_build\\_finish](#) ([atcacert\\_build\\_state\\_t](#) \*build\_state)  
*Completes any final certificate processing required after all data from the device has been incorporated.*
- int [atcacert\\_get\\_device\\_data](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, const [atcacert\\_device\\_loc\\_t](#) \*device\_loc, uint8\_t \*device\_data)  
*Gets the dynamic data that would be saved to the specified device location. This function is primarily used to break down a full certificate into the dynamic components to be saved to a device.*
- int [atcacert\\_set\\_subj\\_public\\_key](#) (const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t cert\_size, const uint8\_t subj\_public\_key[64])  
*Sets the subject public key and subject key ID in a certificate.*
- int [atcacert\\_get\\_subj\\_public\\_key](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t subj\_public\_key[64])  
*Gets the subject public key from a certificate.*
- int [atcacert\\_get\\_subj\\_key\\_id](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t subj\_key\_id[20])  
*Gets the subject key ID from a certificate.*
- int [atcacert\\_set\\_signature](#) (const [atcacert\\_def\\_t](#) \*cert\_def, uint8\_t \*cert, size\_t \*cert\_size, size\_t max\_cert\_size, const uint8\_t signature[64])  
*Sets the signature in a certificate. This may alter the size of the X.509 certificates.*
- int [atcacert\\_get\\_signature](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, uint8\_t signature[64])  
*Gets the signature from a certificate.*

- `int atcacert_set_issue_date (const atcacert_def_t *cert_def, uint8_t *cert, size_t cert_size, const atcacert_tm_utc_t *timestamp)`  
*Sets the issue date (notBefore) in a certificate. Will be formatted according to the date format specified in the certificate definition.*
- `int atcacert_get_issue_date (const atcacert_def_t *cert_def, const uint8_t *cert, size_t cert_size, atcacert_tm_utc_t *timestamp)`  
*Gets the issue date from a certificate. Will be parsed according to the date format specified in the certificate definition.*
- `int atcacert_set_expire_date (const atcacert_def_t *cert_def, uint8_t *cert, size_t cert_size, const atcacert_tm_utc_t *timestamp)`  
*Sets the expire date (notAfter) in a certificate. Will be formatted according to the date format specified in the certificate definition.*
- `int atcacert_get_expire_date (const atcacert_def_t *cert_def, const uint8_t *cert, size_t cert_size, atcacert_tm_utc_t *timestamp)`  
*Gets the expire date from a certificate. Will be parsed according to the date format specified in the certificate definition.*
- `int atcacert_set_signer_id (const atcacert_def_t *cert_def, uint8_t *cert, size_t cert_size, const uint8_t signer_id[2])`  
*Sets the signer ID in a certificate. Will be formatted as 4 upper-case hex digits.*
- `int atcacert_get_signer_id (const atcacert_def_t *cert_def, const uint8_t *cert, size_t cert_size, uint8_t signer_id[2])`  
*Gets the signer ID from a certificate. Will be parsed as 4 upper-case hex digits.*
- `int atcacert_set_cert_sn (const atcacert_def_t *cert_def, uint8_t *cert, size_t *cert_size, size_t max_cert_size, const uint8_t *cert_sn, size_t cert_sn_size)`  
*Sets the certificate serial number in a certificate.*
- `int atcacert_gen_cert_sn (const atcacert_def_t *cert_def, uint8_t *cert, size_t cert_size, const uint8_t device_sn[9])`  
*Sets the certificate serial number by generating it from other information in the certificate using the scheme specified by sn\_source in cert\_def. See the.*
- `int atcacert_get_cert_sn (const atcacert_def_t *cert_def, const uint8_t *cert, size_t cert_size, uint8_t *cert_sn, size_t *cert_sn_size)`  
*Gets the certificate serial number from a certificate.*
- `int atcacert_set_auth_key_id (const atcacert_def_t *cert_def, uint8_t *cert, size_t cert_size, const uint8_t auth_public_key[64])`  
*Sets the authority key ID in a certificate. Note that this takes the actual public key creates a key ID from it.*
- `int atcacert_set_auth_key_id_raw (const atcacert_def_t *cert_def, uint8_t *cert, size_t cert_size, const uint8_t *auth_key_id)`  
*Sets the authority key ID in a certificate.*
- `int atcacert_get_auth_key_id (const atcacert_def_t *cert_def, const uint8_t *cert, size_t cert_size, uint8_t auth_key_id[20])`  
*Gets the authority key ID from a certificate.*
- `int atcacert_set_comp_cert (const atcacert_def_t *cert_def, uint8_t *cert, size_t *cert_size, size_t max_cert_size, const uint8_t comp_cert[72])`  
*Sets the signature, issue date, expire date, and signer ID found in the compressed certificate. This also checks fields common between the cert\_def and the compressed certificate to make sure they match.*
- `int atcacert_get_comp_cert (const atcacert_def_t *cert_def, const uint8_t *cert, size_t cert_size, uint8_t comp_cert[72])`  
*Generate the compressed certificate for the given certificate.*
- `int atcacert_get_tbs (const atcacert_def_t *cert_def, const uint8_t *cert, size_t cert_size, const uint8_t **tbs, size_t *tbs_size)`  
*Get a pointer to the TBS data in a certificate.*
- `int atcacert_get_tbs_digest (const atcacert_def_t *cert_def, const uint8_t *cert, size_t cert_size, uint8_t tbs_digest[32])`  
*Get the SHA256 digest of certificate's TBS data.*
- `int atcacert_set_cert_element (const atcacert_def_t *cert_def, const atcacert_cert_loc_t *cert_loc, uint8_t *cert, size_t cert_size, const uint8_t *data, size_t data_size)`

*Sets an element in a certificate. The data\_size must match the size in cert\_loc.*

- int [atcacert\\_get\\_cert\\_element](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const [atcacert\\_cert\\_loc\\_t](#) \*cert\_loc, const uint8\_t \*cert, size\_t cert\_size, uint8\_t \*data, size\_t data\_size)

*Gets an element from a certificate.*

- int [atcacert\\_get\\_key\\_id](#) (const uint8\_t public\_key[64], uint8\_t key\_id[20])

*Calculates the key ID for a given public ECC P256 key.*

- int [atcacert\\_merge\\_device\\_loc](#) ([atcacert\\_device\\_loc\\_t](#) \*device\_locs, size\_t \*device\_locs\_count, size\_t device\_locs\_max\_count, const [atcacert\\_device\\_loc\\_t](#) \*device\_loc, size\_t block\_size)

*Merge a new device location into a list of device locations. If the new location overlaps with an existing location, the existing one will be modified to encompass both. Otherwise the new location is appended to the end of the list.*

- int [atcacert\\_is\\_device\\_loc\\_overlap](#) (const [atcacert\\_device\\_loc\\_t](#) \*device\_loc1, const [atcacert\\_device\\_loc\\_t](#) \*device\_loc2)

*Determines if the two device locations overlap.*

- void [atcacert\\_public\\_key\\_add\\_padding](#) (const uint8\_t raw\_key[64], uint8\_t padded\_key[72])

*Takes a raw P256 ECC public key and converts it to the padded version used by ATECC devices. Input and output buffers can point to the same location to do an in-place transform.*

- void [atcacert\\_public\\_key\\_remove\\_padding](#) (const uint8\_t padded\_key[72], uint8\_t raw\_key[64])

*Takes a padded public key used by ATECC devices and converts it to a raw P256 ECC public key. Input and output buffers can point to the same location to do an in-place transform.*

- int [atcacert\\_transform\\_data](#) ([atcacert\\_transform\\_t](#) transform, const uint8\_t \*data, size\_t data\_size, uint8\_t \*destination, size\_t \*destination\_size)

*Apply the specified transform to the specified data.*

- int [atcacert\\_max\\_cert\\_size](#) (const [atcacert\\_def\\_t](#) \*cert\_def, size\_t \*max\_cert\_size)

*Return the maximum possible certificate size in bytes for a given cert def. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificates.*

## 20.55.1 Detailed Description

Declarations for certificates related to ECC CryptoAuthentication devices. These are the definitions required to define a certificate and its various elements with regards to the CryptoAuthentication ECC devices.

Only the dynamic elements of a certificate (the parts of the certificate that change from device to device) are stored on the ATECC device. The definitions here describe the form of the certificate, and where the dynamic elements can be found both on the ATECC device itself and in the certificate template.

This also defines utility functions for working with the certificates and their definitions.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.55.2 Macro Definition Documentation

### 20.55.2.1 ATCA\_MAX\_TRANSFORMS

```
#define ATCA_MAX_TRANSFORMS 2
```

## 20.56 atcacert\_der.c File Reference

functions required to work with DER encoded data related to X.509 certificates.

```
#include "atcacert_der.h"
#include <string.h>
```

### Functions

- int [atcacert\\_der\\_enc\\_length](#) (uint32\_t length, uint8\_t \*der\_length, size\_t \*der\_length\_size)  
*Encode a length in DER format.*
- int [atcacert\\_der\\_dec\\_length](#) (const uint8\_t \*der\_length, size\_t \*der\_length\_size, uint32\_t \*length)  
*Decode a DER format length.*
- int [atcacert\\_der\\_adjust\\_length](#) (uint8\_t \*der\_length, size\_t \*der\_length\_size, int delta\_length, uint32\_t \*new\_length)
- int [atcacert\\_der\\_enc\\_integer](#) (const uint8\_t \*int\_data, size\_t int\_data\_size, uint8\_t is\_unsigned, uint8\_t \*der\_int, size\_t \*der\_int\_size)  
*Encode an ASN.1 integer in DER format, including tag and length fields.*
- int [atcacert\\_der\\_dec\\_integer](#) (const uint8\_t \*der\_int, size\_t \*der\_int\_size, uint8\_t \*int\_data, size\_t \*int\_data\_size)  
*Decode an ASN.1 DER encoded integer.*
- int [atcacert\\_der\\_enc\\_ecdsa\\_sig\\_value](#) (const uint8\_t raw\_sig[64], uint8\_t \*der\_sig, size\_t \*der\_sig\_size)  
*Formats a raw ECDSA P256 signature in the DER encoding found in X.509 certificates.*
- int [atcacert\\_der\\_dec\\_ecdsa\\_sig\\_value](#) (const uint8\_t \*der\_sig, size\_t \*der\_sig\_size, uint8\_t raw\_sig[64])  
*Parses an ECDSA P256 signature in the DER encoding as found in X.509 certificates.*

### 20.56.1 Detailed Description

functions required to work with DER encoded data related to X.509 certificates.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.57 atcacert\_der.h File Reference

function declarations required to work with DER encoded data related to X.509 certificates.

```
#include <stddef.h>
#include <stdint.h>
#include "atcacert.h"
```

## Functions

- int [atcacert\\_der\\_enc\\_length](#) (uint32\_t length, uint8\_t \*der\_length, size\_t \*der\_length\_size)  
*Encode a length in DER format.*
- int [atcacert\\_der\\_dec\\_length](#) (const uint8\_t \*der\_length, size\_t \*der\_length\_size, uint32\_t \*length)  
*Decode a DER format length.*
- int [atcacert\\_der\\_adjust\\_length](#) (uint8\_t \*der\_length, size\_t \*der\_length\_size, int delta\_length, uint32\_t \*new\_length)
- int [atcacert\\_der\\_enc\\_integer](#) (const uint8\_t \*int\_data, size\_t int\_data\_size, uint8\_t is\_unsigned, uint8\_t \*der\_int, size\_t \*der\_int\_size)  
*Encode an ASN.1 integer in DER format, including tag and length fields.*
- int [atcacert\\_der\\_dec\\_integer](#) (const uint8\_t \*der\_int, size\_t \*der\_int\_size, uint8\_t \*int\_data, size\_t \*int\_data\_size)  
*Decode an ASN.1 DER encoded integer.*
- int [atcacert\\_der\\_enc\\_ecdsa\\_sig\\_value](#) (const uint8\_t raw\_sig[64], uint8\_t \*der\_sig, size\_t \*der\_sig\_size)  
*Formats a raw ECDSA P256 signature in the DER encoding found in X.509 certificates.*
- int [atcacert\\_der\\_dec\\_ecdsa\\_sig\\_value](#) (const uint8\_t \*der\_sig, size\_t \*der\_sig\_size, uint8\_t raw\_sig[64])  
*Parses an ECDSA P256 signature in the DER encoding as found in X.509 certificates.*

### 20.57.1 Detailed Description

function declarations required to work with DER encoded data related to X.509 certificates.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.58 atcacert\_host\_hw.c File Reference

host side methods using CryptoAuth hardware

```
#include "atcacert_host_hw.h"
#include "atca_basic.h"
#include "crypto/atca_crypto_sw_sha2.h"
```

## Functions

- int [atcacert\\_verify\\_cert\\_hw](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, const uint8\_t ca\_public\_key[64])  
*Verify a certificate against its certificate authority's public key using the host's ATECC device for crypto functions.*
- int [atcacert\\_gen\\_challenge\\_hw](#) (uint8\_t challenge[32])  
*Generate a random challenge to be sent to the client using the RNG on the host's ATECC device.*
- int [atcacert\\_verify\\_response\\_hw](#) (const uint8\_t device\_public\_key[64], const uint8\_t challenge[32], const uint8\_t response[64])  
*Verify a client's response to a challenge using the host's ATECC device for crypto functions.*



### 20.58.1 Detailed Description

host side methods using CryptoAuth hardware

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.59 atcacert\_host\_hw.h File Reference

host side methods using CryptoAuth hardware

```
#include <stddef.h>
#include <stdint.h>
#include "atcacert_def.h"
```

### Functions

- int [atcacert\\_verify\\_cert\\_hw](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, const uint8\_t ca\_public\_key[64])  
*Verify a certificate against its certificate authority's public key using the host's ATECC device for crypto functions.*
- int [atcacert\\_gen\\_challenge\\_hw](#) (uint8\_t challenge[32])  
*Generate a random challenge to be sent to the client using the RNG on the host's ATECC device.*
- int [atcacert\\_verify\\_response\\_hw](#) (const uint8\_t device\_public\_key[64], const uint8\_t challenge[32], const uint8\_t response[64])  
*Verify a client's response to a challenge using the host's ATECC device for crypto functions.*

### 20.59.1 Detailed Description

host side methods using CryptoAuth hardware

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.60 atcacert\_host\_sw.c File Reference

host side methods using software implementations

```
#include "atcacert_host_sw.h"
#include "crypto/atca_crypto_sw_sha2.h"
#include "crypto/atca_crypto_sw_ecdsa.h"
#include "crypto/atca_crypto_sw_rand.h"
```

### Functions

- int [atcacert\\_verify\\_cert\\_sw](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, const uint8\_t ca\_public\_key[64])  
*Verify a certificate against its certificate authority's public key using software crypto functions. The function is currently not implemented.*
- int [atcacert\\_gen\\_challenge\\_sw](#) (uint8\_t challenge[32])  
*Generate a random challenge to be sent to the client using a software PRNG. The function is currently not implemented.*
- int [atcacert\\_verify\\_response\\_sw](#) (const uint8\_t device\_public\_key[64], const uint8\_t challenge[32], const uint8\_t response[64])  
*Verify a client's response to a challenge using software crypto functions. The function is currently not implemented.*

#### 20.60.1 Detailed Description

host side methods using software implementations

##### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.61 atcacert\_host\_sw.h File Reference

Host side methods using software implementations. host-side, the one authenticating a client, of the authentication process. Crypto functions are performed using a software library.

```
#include <stddef.h>
#include <stdint.h>
#include "atcacert_def.h"
```

### Functions

- int [atcacert\\_verify\\_cert\\_sw](#) (const [atcacert\\_def\\_t](#) \*cert\_def, const uint8\_t \*cert, size\_t cert\_size, const uint8\_t ca\_public\_key[64])  
*Verify a certificate against its certificate authority's public key using software crypto functions. The function is currently not implemented.*
- int [atcacert\\_gen\\_challenge\\_sw](#) (uint8\_t challenge[32])  
*Generate a random challenge to be sent to the client using a software PRNG. The function is currently not implemented.*
- int [atcacert\\_verify\\_response\\_sw](#) (const uint8\_t device\_public\_key[64], const uint8\_t challenge[32], const uint8\_t response[64])  
*Verify a client's response to a challenge using software crypto functions. The function is currently not implemented.*

#### 20.61.1 Detailed Description

Host side methods using software implementations. host-side, the one authenticating a client, of the authentication process. Crypto functions are performed using a software library.

##### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.62 atcacert\_pem.c File Reference

Functions required to work with PEM encoded data related to X.509 certificates.

```
#include <string.h>
#include "atcacert.h"
#include "atcacert_pem.h"
#include "atca_helpers.h"
```

### Functions

- int [atcacert\\_encode\\_pem](#) (const uint8\_t \*der, size\_t der\_size, char \*pem, size\_t \*pem\_size, const char \*header, const char \*footer)  
*Encode a DER data in PEM format.*
- int [atcacert\\_decode\\_pem](#) (const char \*pem, size\_t pem\_size, uint8\_t \*der, size\_t \*der\_size, const char \*header, const char \*footer)  
*Decode PEM data into DER format.*
- int [atcacert\\_encode\\_pem\\_cert](#) (const uint8\_t \*der\_cert, size\_t der\_cert\_size, char \*pem\_cert, size\_t \*pem\_cert\_size)  
*Encode a DER certificate in PEM format.*
- int [atcacert\\_encode\\_pem\\_csr](#) (const uint8\_t \*der\_csr, size\_t der\_csr\_size, char \*pem\_csr, size\_t \*pem\_csr\_size)  
*Encode a DER CSR in PEM format.*
- int [atcacert\\_decode\\_pem\\_cert](#) (const char \*pem\_cert, size\_t pem\_cert\_size, uint8\_t \*der\_cert, size\_t \*der\_cert\_size)  
*Decode a PEM certificate into DER format.*
- int [atcacert\\_decode\\_pem\\_csr](#) (const char \*pem\_csr, size\_t pem\_csr\_size, uint8\_t \*der\_csr, size\_t \*der\_csr\_size)  
*Extract the CSR certificate bytes from a PEM encoded CSR certificate.*

### 20.62.1 Detailed Description

Functions required to work with PEM encoded data related to X.509 certificates.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.62.2 Function Documentation

#### 20.62.2.1 atcacert\_decode\_pem()

```
int atcacert_decode_pem (
    const char * pem,
    size_t pem_size,
    uint8_t * der,
    size_t * der_size,
    const char * header,
    const char * footer )
```

Decode PEM data into DER format.

### Parameters

in	<i>pem</i>	PEM data to decode to DER.
in	<i>pem_size</i>	PEM data size in bytes.
out	<i>der</i>	DER data is returned here.
in, out	<i>der_size</i>	As input, the size of the der buffer. As output, the size of the DER data.
in	<i>header</i>	Header to find the beginning of the PEM data.
in	<i>footer</i>	Footer to find the end of the PEM data.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.62.2.2 atcacert\_decode\_pem\_cert()

```
int atcacert_decode_pem_cert (
    const char * pem_cert,
    size_t pem_cert_size,
    uint8_t * der_cert,
    size_t * der_cert_size )
```

Decode a PEM certificate into DER format.

### Parameters

in	<i>pem_cert</i>	PEM certificate to decode to DER.
in	<i>pem_cert_size</i>	PEM certificate size in bytes.
out	<i>der_cert</i>	DER certificate is returned here.
in, out	<i>der_cert_size</i>	As input, the size of the der_cert buffer. As output, the size of the DER certificate.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.62.2.3 atcacert\_decode\_pem\_csr()

```
int atcacert_decode_pem_csr (
    const char * pem_csr,
    size_t pem_csr_size,
    uint8_t * der_csr,
    size_t * der_csr_size )
```

Extract the CSR certificate bytes from a PEM encoded CSR certificate.

**Parameters**

in	<i>pem_csr</i>	PEM CSR to decode to DER.
in	<i>pem_csr_size</i>	PEM CSR size in bytes.
out	<i>der_csr</i>	DER CSR is returned here.
in, out	<i>der_csr_size</i>	As input, the size of the der_csr buffer. As output, the size of the DER CSR.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.62.2.4 atcacert\_encode\_pem()**

```
int atcacert_encode_pem (
    const uint8_t * der,
    size_t der_size,
    char * pem,
    size_t * pem_size,
    const char * header,
    const char * footer )
```

Encode a DER data in PEM format.

**Parameters**

in	<i>der</i>	DER data to be encoded as PEM.
out	<i>der_size</i>	DER data size in bytes.
out	<i>pem</i>	PEM encoded data is returned here.
in, out	<i>pem_size</i>	As input, the size of the pem buffer. As output, the size of the PEM data.
in	<i>header</i>	Header to place at the beginning of the PEM data.
in	<i>footer</i>	Footer to place at the end of the PEM data.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.62.2.5 atcacert\_encode\_pem\_cert()**

```
int atcacert_encode_pem_cert (
    const uint8_t * der_cert,
    size_t der_cert_size,
    char * pem_cert,
    size_t * pem_cert_size )
```

Encode a DER certificate in PEM format.

## 20.63 atcacert\_pem.h File Reference

---

### Parameters

in	<i>der_cert</i>	DER certificate to be encoded as PEM.
out	<i>der_cert_size</i>	DER certificate size in bytes.
out	<i>pem_cert</i>	PEM encoded certificate is returned here.
in, out	<i>pem_cert_size</i>	As input, the size of the pem_cert buffer. As output, the size of the PEM certificate.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.62.2.6 atcacert\_encode\_pem\_csr()

```
int atcacert_encode_pem_csr (
    const uint8_t * der_csr,
    size_t der_csr_size,
    char * pem_csr,
    size_t * pem_csr_size )
```

Encode a DER CSR in PEM format.

### Parameters

in	<i>der_csr</i>	DER CSR to be encoded as PEM.
out	<i>der_csr_size</i>	DER CSR size in bytes.
out	<i>pem_csr</i>	PEM encoded CSR is returned here.
in, out	<i>pem_csr_size</i>	As input, the size of the pem_csr buffer. As output, the size of the PEM CSR.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.63 atcacert\_pem.h File Reference

Functions for converting between DER and PEM formats.

```
#include <stdint.h>
```

### Macros

- #define [PEM\\_CERT\\_BEGIN](#) "-----BEGIN CERTIFICATE-----"
- #define [PEM\\_CERT\\_END](#) "-----END CERTIFICATE-----"
- #define [PEM\\_CSR\\_BEGIN](#) "-----BEGIN CERTIFICATE REQUEST-----"
- #define [PEM\\_CSR\\_END](#) "-----END CERTIFICATE REQUEST-----"

## Functions

- int [atcacert\\_encode\\_pem](#) (const uint8\_t \*der, size\_t der\_size, char \*pem, size\_t \*pem\_size, const char \*header, const char \*footer)  
*Encode a DER data in PEM format.*
- int [atcacert\\_decode\\_pem](#) (const char \*pem, size\_t pem\_size, uint8\_t \*der, size\_t \*der\_size, const char \*header, const char \*footer)  
*Decode PEM data into DER format.*
- int [atcacert\\_encode\\_pem\\_cert](#) (const uint8\_t \*der\_cert, size\_t der\_cert\_size, char \*pem\_cert, size\_t \*pem\_cert\_size)  
*Encode a DER certificate in PEM format.*
- int [atcacert\\_decode\\_pem\\_cert](#) (const char \*pem\_cert, size\_t pem\_cert\_size, uint8\_t \*der\_cert, size\_t \*der\_cert\_size)  
*Decode a PEM certificate into DER format.*
- int [atcacert\\_encode\\_pem\\_csr](#) (const uint8\_t \*der\_csr, size\_t der\_csr\_size, char \*pem\_csr, size\_t \*pem\_csr\_size)  
*Encode a DER CSR in PEM format.*
- int [atcacert\\_decode\\_pem\\_csr](#) (const char \*pem\_csr, size\_t pem\_csr\_size, uint8\_t \*der\_csr, size\_t \*der\_csr\_size)  
*Extract the CSR certificate bytes from a PEM encoded CSR certificate.*

### 20.63.1 Detailed Description

Functions for converting between DER and PEM formats.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.63.2 Macro Definition Documentation

#### 20.63.2.1 PEM\_CERT\_BEGIN

```
#define PEM_CERT_BEGIN "-----BEGIN CERTIFICATE-----"
```

#### 20.63.2.2 PEM\_CERT\_END

```
#define PEM_CERT_END "-----END CERTIFICATE-----"
```

### 20.63.2.3 PEM\_CSR\_BEGIN

```
#define PEM_CSR_BEGIN "-----BEGIN CERTIFICATE REQUEST-----"
```

### 20.63.2.4 PEM\_CSR\_END

```
#define PEM_CSR_END "-----END CERTIFICATE REQUEST-----"
```

## 20.63.3 Function Documentation

### 20.63.3.1 atcacert\_decode\_pem()

```
int atcacert_decode_pem (  
    const char * pem,  
    size_t pem_size,  
    uint8_t * der,  
    size_t * der_size,  
    const char * header,  
    const char * footer )
```

Decode PEM data into DER format.

#### Parameters

in	<i>pem</i>	PEM data to decode to DER.
in	<i>pem_size</i>	PEM data size in bytes.
out	<i>der</i>	DER data is returned here.
in, out	<i>der_size</i>	As input, the size of the der buffer. As output, the size of the DER data.
in	<i>header</i>	Header to find the beginning of the PEM data.
in	<i>footer</i>	Footer to find the end of the PEM data.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.63.3.2 atcacert\_decode\_pem\_cert()

```
int atcacert_decode_pem_cert (  
    const char * pem_cert,  
    size_t pem_cert_size,  
    uint8_t * der_cert,  
    size_t * der_cert_size )
```

Decode a PEM certificate into DER format.



**Parameters**

in	<i>pem_cert</i>	PEM certificate to decode to DER.
in	<i>pem_cert_size</i>	PEM certificate size in bytes.
out	<i>der_cert</i>	DER certificate is returned here.
in, out	<i>der_cert_size</i>	As input, the size of the der_cert buffer. As output, the size of the DER certificate.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.63.3.3 atcacert\_decode\_pem\_csr()**

```
int atcacert_decode_pem_csr (
    const char * pem_csr,
    size_t pem_csr_size,
    uint8_t * der_csr,
    size_t * der_csr_size )
```

Extract the CSR certificate bytes from a PEM encoded CSR certificate.

**Parameters**

in	<i>pem_csr</i>	PEM CSR to decode to DER.
in	<i>pem_csr_size</i>	PEM CSR size in bytes.
out	<i>der_csr</i>	DER CSR is returned here.
in, out	<i>der_csr_size</i>	As input, the size of the der_csr buffer. As output, the size of the DER CSR.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.63.3.4 atcacert\_encode\_pem()**

```
int atcacert_encode_pem (
    const uint8_t * der,
    size_t der_size,
    char * pem,
    size_t * pem_size,
    const char * header,
    const char * footer )
```

Encode a DER data in PEM format.

### Parameters

in	<i>der</i>	DER data to be encoded as PEM.
out	<i>der_size</i>	DER data size in bytes.
out	<i>pem</i>	PEM encoded data is returned here.
in, out	<i>pem_size</i>	As input, the size of the pem buffer. As output, the size of the PEM data.
in	<i>header</i>	Header to place at the beginning of the PEM data.
in	<i>footer</i>	Footer to place at the end of the PEM data.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.63.3.5 atcacert\_encode\_pem\_cert()

```
int atcacert_encode_pem_cert (
    const uint8_t * der_cert,
    size_t der_cert_size,
    char * pem_cert,
    size_t * pem_cert_size )
```

Encode a DER certificate in PEM format.

### Parameters

in	<i>der_cert</i>	DER certificate to be encoded as PEM.
out	<i>der_cert_size</i>	DER certificate size in bytes.
out	<i>pem_cert</i>	PEM encoded certificate is returned here.
in, out	<i>pem_cert_size</i>	As input, the size of the pem_cert buffer. As output, the size of the PEM certificate.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.63.3.6 atcacert\_encode\_pem\_csr()

```
int atcacert_encode_pem_csr (
    const uint8_t * der_csr,
    size_t der_csr_size,
    char * pem_csr,
    size_t * pem_csr_size )
```

Encode a DER CSR in PEM format.

**Parameters**

in	<i>der_csr</i>	DER CSR to be encoded as PEM.
out	<i>der_csr_size</i>	DER CSR size in bytes.
out	<i>pem_csr</i>	PEM encoded CSR is returned here.
in, out	<i>pem_csr_size</i>	As input, the size of the pem_csr buffer. As output, the size of the PEM CSR.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

## 20.64 calib\_aes.c File Reference

CryptoAuthLib Basic API methods for AES command.

```
#include "cryptoauthlib.h"
```

**Functions**

- [ATCA\\_STATUS calib\\_aes](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*aes\_in, uint8\_t \*aes\_out)  
*Compute the AES-128 encrypt, decrypt, or GFM calculation.*
- [ATCA\\_STATUS calib\\_aes\\_encrypt](#) ([ATCADevice](#) device, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*plaintext, uint8\_t \*ciphertext)  
*Perform an AES-128 encrypt operation with a key in the device.*
- [ATCA\\_STATUS calib\\_aes\\_decrypt](#) ([ATCADevice](#) device, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*ciphertext, uint8\_t \*plaintext)  
*Perform an AES-128 decrypt operation with a key in the device.*
- [ATCA\\_STATUS calib\\_aes\\_gfm](#) ([ATCADevice](#) device, const uint8\_t \*h, const uint8\_t \*input, uint8\_t \*output)  
*Perform a Galois Field Multiply (GFM) operation.*

### 20.64.1 Detailed Description

CryptoAuthLib Basic API methods for AES command.

The AES command supports 128-bit AES encryption or decryption of small messages or data packets in ECB mode. Also can perform GFM (Galois Field Multiply) calculation in support of AES-GCM.

**Note**

List of devices that support this command - ATECC608A/B. Refer to device edatasheet for full details.

**Copyright**

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.65 calib\_aes\_gcm.c File Reference

CryptoAuthLib Basic API methods for AES GCM mode.

```
#include "cryptoauthlib.h"
#include "calib_aes_gcm.h"
```

- `#define RETURN` return `ATCA_TRACE`
- `const char * atca_basic_aes_gcm_version = "2.0"`
- `ATCA_STATUS calib_aes_gcm_init` (`ATCADevice` device, `atca_aes_gcm_ctx_t *ctx`, `uint16_t key_id`, `uint8_t key_block`, `const uint8_t *iv`, `size_t iv_size`)  
*Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.*
- `ATCA_STATUS calib_aes_gcm_init_rand` (`ATCADevice` device, `atca_aes_gcm_ctx_t *ctx`, `uint16_t key_id`, `uint8_t key_block`, `size_t rand_size`, `const uint8_t *free_field`, `size_t free_field_size`, `uint8_t *iv`)  
*Initialize context for AES GCM operation with a IV composed of a random and optional fixed(free) field, which is common when starting an encrypt operation.*
- `ATCA_STATUS calib_aes_gcm_aad_update` (`ATCADevice` device, `atca_aes_gcm_ctx_t *ctx`, `const uint8_t *aad`, `uint32_t aad_size`)  
*Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.*
- `ATCA_STATUS calib_aes_gcm_encrypt_update` (`ATCADevice` device, `atca_aes_gcm_ctx_t *ctx`, `const uint8_t *plaintext`, `uint32_t plaintext_size`, `uint8_t *ciphertext`)  
*Encrypt data using GCM mode and a key within the ATECC608 device. `atcab_aes_gcm_init()` or `atcab_aes_gcm_init_rand()` should be called before the first use of this function.*
- `ATCA_STATUS calib_aes_gcm_encrypt_finish` (`ATCADevice` device, `atca_aes_gcm_ctx_t *ctx`, `uint8_t *tag`, `size_t tag_size`)  
*Complete a GCM encrypt operation returning the authentication tag.*
- `ATCA_STATUS calib_aes_gcm_decrypt_update` (`ATCADevice` device, `atca_aes_gcm_ctx_t *ctx`, `const uint8_t *ciphertext`, `uint32_t ciphertext_size`, `uint8_t *plaintext`)  
*Decrypt data using GCM mode and a key within the ATECC608 device. `atcab_aes_gcm_init()` or `atcab_aes_gcm_init_rand()` should be called before the first use of this function.*
- `ATCA_STATUS calib_aes_gcm_decrypt_finish` (`ATCADevice` device, `atca_aes_gcm_ctx_t *ctx`, `const uint8_t *tag`, `size_t tag_size`, `bool *is_verified`)  
*Complete a GCM decrypt operation verifying the authentication tag.*

### 20.65.1 Detailed Description

CryptoAuthLib Basic API methods for AES GCM mode.

The AES command supports 128-bit AES encryption or decryption of small messages or data packets in ECB mode. Also can perform GFM (Galois Field Multiply) calculation in support of AES-GCM.

#### Note

List of devices that support this command - ATECC608A/B. Refer to device datasheet for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.65.2 Macro Definition Documentation

### 20.65.2.1 RETURN

```
#define RETURN return ATCA_TRACE
```

## 20.65.3 Function Documentation

### 20.65.3.1 calib\_aes\_gcm\_aad\_update()

```
ATCA_STATUS calib_aes_gcm_aad_update (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * aad,
    uint32_t aad_size )
```

Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.

This can be called multiple times. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function. When there is AAD to include, this should be called before [atcab\\_aes\\_gcm\\_encrypt\\_update\(\)](#) or [atcab\\_aes\\_gcm\\_decrypt\\_update\(\)](#).

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES GCM context
in	<i>aad</i>	Additional authenticated data to be added
in	<i>aad_size</i>	Size of aad in bytes

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.65.3.2 calib\_aes\_gcm\_decrypt\_finish()

```
ATCA_STATUS calib_aes_gcm_decrypt_finish (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * tag,
    size_t tag_size,
    bool * is_verified )
```

Complete a GCM decrypt operation verifying the authentication tag.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES GCM context structure.
in	<i>tag</i>	Expected authentication tag.
in	<i>tag_size</i>	Size of tag in bytes (12 to 16 bytes).
out	<i>is_verified</i>	Returns whether or not the tag verified.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.65.3.3 calib\_aes\_gcm\_decrypt\_update()**

```
ATCA_STATUS calib_aes_gcm_decrypt_update (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * ciphertext,
    uint32_t ciphertext_size,
    uint8_t * plaintext )
```

Decrypt data using GCM mode and a key within the ATECC608 device. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function.

**Parameters**

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES GCM context structure.
in	<i>ciphertext</i>	Ciphertext to be decrypted.
in	<i>ciphertext_size</i>	Size of ciphertext in bytes.
out	<i>plaintext</i>	Decrypted data is returned here.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.65.3.4 calib\_aes\_gcm\_encrypt\_finish()**

```
ATCA_STATUS calib_aes_gcm_encrypt_finish (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    uint8_t * tag,
    size_t tag_size )
```

Complete a GCM encrypt operation returning the authentication tag.

## Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES GCM context structure.
out	<i>tag</i>	Authentication tag is returned here.
in	<i>tag_size</i>	Tag size in bytes (12 to 16 bytes).

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.65.3.5 calib\_aes\_gcm\_encrypt\_update()

```
ATCA_STATUS calib_aes_gcm_encrypt_update (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    const uint8_t * plaintext,
    uint32_t plaintext_size,
    uint8_t * ciphertext )
```

Encrypt data using GCM mode and a key within the ATECC608 device. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function.

## Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES GCM context structure.
in	<i>plaintext</i>	Plaintext to be encrypted (16 bytes).
in	<i>plaintext_size</i>	Size of plaintext in bytes.
out	<i>ciphertext</i>	Encrypted data is returned here.

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.65.3.6 calib\_aes\_gcm\_init()

```
ATCA_STATUS calib_aes_gcm_init (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block,
    const uint8_t * iv,
    size_t iv_size )
```

Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.

## 20.66 calib\_aes\_gcm.h File Reference

### Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES GCM context to be initialized.
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>iv</i>	Initialization vector.
in	<i>iv_size</i>	Size of IV in bytes. Standard is 12 bytes.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.65.3.7 calib\_aes\_gcm\_init\_rand()

```
ATCA_STATUS calib_aes_gcm_init_rand (
    ATCADevice device,
    atca_aes_gcm_ctx_t * ctx,
    uint16_t key_id,
    uint8_t key_block,
    size_t rand_size,
    const uint8_t * free_field,
    size_t free_field_size,
    uint8_t * iv )
```

Initialize context for AES GCM operation with a IV composed of a random and optional fixed(free) field, which is common when starting an encrypt operation.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>ctx</i>	AES CTR context to be initialized.
in	<i>key_id</i>	Key location. Can either be a slot number or ATCA_TEMPKEY_KEYID for TempKey.
in	<i>key_block</i>	Index of the 16-byte block to use within the key location for the actual key.
in	<i>rand_size</i>	Size of the random field in bytes. Minimum and recommended size is 12 bytes. Max is 32 bytes.
in	<i>free_field</i>	Fixed data to include in the IV after the random field. Can be NULL if not used.
in	<i>free_field_size</i>	Size of the free field in bytes.
out	<i>iv</i>	Initialization vector is returned here. Its size will be rand_size and free_field_size combined.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.66 calib\_aes\_gcm.h File Reference

Unity tests for the cryptoauthlib AES GCM functions.



## Data Structures

- struct [atca\\_aes\\_gcm\\_ctx](#)
  
- #define [ATCA\\_AES\\_GCM\\_IV\\_STD\\_LENGTH](#) 12
- typedef struct [atca\\_aes\\_gcm\\_ctx](#) [atca\\_aes\\_gcm\\_ctx\\_t](#)
- const char \* [atca\\_basic\\_aes\\_gcm\\_version](#)
- [ATCA\\_STATUS](#) [calib\\_aes\\_gcm\\_init](#) ([ATCADevice](#) device, [atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*iv, size\_t iv\_size)  
*Initialize context for AES GCM operation with an existing IV, which is common when starting a decrypt operation.*
- [ATCA\\_STATUS](#) [calib\\_aes\\_gcm\\_init\\_rand](#) ([ATCADevice](#) device, [atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, uint16\_t key\_id, uint8\_t key\_block, size\_t rand\_size, const uint8\_t \*free\_field, size\_t free\_field\_size, uint8\_t \*iv)  
*Initialize context for AES GCM operation with a IV composed of a random and optional fixed(free) field, which is common when starting an encrypt operation.*
- [ATCA\\_STATUS](#) [calib\\_aes\\_gcm\\_aad\\_update](#) ([ATCADevice](#) device, [atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*aad, uint32\_t aad\_size)  
*Process Additional Authenticated Data (AAD) using GCM mode and a key within the ATECC608 device.*
- [ATCA\\_STATUS](#) [calib\\_aes\\_gcm\\_encrypt\\_update](#) ([ATCADevice](#) device, [atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*plaintext, uint32\_t plaintext\_size, uint8\_t \*ciphertext)  
*Encrypt data using GCM mode and a key within the ATECC608 device. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function.*
- [ATCA\\_STATUS](#) [calib\\_aes\\_gcm\\_encrypt\\_finish](#) ([ATCADevice](#) device, [atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, uint8\_t \*tag, size\_t tag\_size)  
*Complete a GCM encrypt operation returning the authentication tag.*
- [ATCA\\_STATUS](#) [calib\\_aes\\_gcm\\_decrypt\\_update](#) ([ATCADevice](#) device, [atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*ciphertext, uint32\_t ciphertext\_size, uint8\_t \*plaintext)  
*Decrypt data using GCM mode and a key within the ATECC608 device. [atcab\\_aes\\_gcm\\_init\(\)](#) or [atcab\\_aes\\_gcm\\_init\\_rand\(\)](#) should be called before the first use of this function.*
- [ATCA\\_STATUS](#) [calib\\_aes\\_gcm\\_decrypt\\_finish](#) ([ATCADevice](#) device, [atca\\_aes\\_gcm\\_ctx\\_t](#) \*ctx, const uint8\_t \*tag, size\_t tag\_size, bool \*is\_verified)  
*Complete a GCM decrypt operation verifying the authentication tag.*

### 20.66.1 Detailed Description

Unity tests for the cryptoauthlib AES GCM functions.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.67 calib\_basic.c File Reference

CryptoAuthLib Basic API methods. These methods provide a simpler way to access the core crypto methods.

```
#include "cryptoauthlib.h"
```

#### Macros

- #define [MAX\\_BUSES](#) 4

### Functions

- [ATCA\\_STATUS calib\\_wakeup](#) ([ATCADevice](#) device)  
*basic API methods are all prefixed with atcab\_ (CryptoAuthLib Basic) the fundamental premise of the basic API is it is based on a single interface instance and that instance is global, so all basic API commands assume that one global device is the one to operate on.*
- [ATCA\\_STATUS calib\\_idle](#) ([ATCADevice](#) device)  
*idle the CryptoAuth device*
- [ATCA\\_STATUS calib\\_sleep](#) ([ATCADevice](#) device)  
*invoke sleep on the CryptoAuth device*
- [ATCA\\_STATUS calib\\_cfg\\_discover](#) ([ATCAIfaceCfg](#) cfg\_array[], int max\_ifaces)  
*auto discovery of crypto auth devices*
- [ATCA\\_STATUS \\_calib\\_exit](#) ([ATCADevice](#) device)  
*common cleanup code which idles the device after any operation*
- [ATCA\\_STATUS calib\\_get\\_addr](#) (uint8\_t zone, uint16\_t slot, uint8\_t block, uint8\_t offset, uint16\_t \*addr)  
*Compute the address given the zone, slot, block, and offset.*
- [ATCA\\_STATUS calib\\_get\\_zone\\_size](#) ([ATCADevice](#) device, uint8\_t zone, uint16\_t slot, size\_t \*size)  
*Gets the size of the specified zone in bytes.*

### 20.67.1 Detailed Description

CryptoAuthLib Basic API methods. These methods provide a simpler way to access the core crypto methods.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.67.2 Macro Definition Documentation

#### 20.67.2.1 MAX\_BUSES

```
#define MAX_BUSES 4
```

## 20.68 calib\_basic.h File Reference

```
#include "calib_command.h"  
#include "calib_execution.h"
```

### Data Structures

- struct [atca\\_sha256\\_ctx](#)

## Typedefs

- typedef struct [atca\\_sha256\\_ctx](#) [atca\\_sha256\\_ctx\\_t](#)
- typedef [atca\\_sha256\\_ctx\\_t](#) [atca\\_hmac\\_sha256\\_ctx\\_t](#)

## Functions

- [ATCA\\_STATUS\\_calib\\_wakeup](#) ([ATCADevice](#) device)  
*basic API methods are all prefixed with atcab\_ (CryptoAuthLib Basic) the fundamental premise of the basic API is it is based on a single interface instance and that instance is global, so all basic API commands assume that one global device is the one to operate on.*
- [ATCA\\_STATUS\\_calib\\_idle](#) ([ATCADevice](#) device)  
*idle the CryptoAuth device*
- [ATCA\\_STATUS\\_calib\\_sleep](#) ([ATCADevice](#) device)  
*invoke sleep on the CryptoAuth device*
- [ATCA\\_STATUS\\_calib\\_exit](#) ([ATCADevice](#) device)  
*common cleanup code which idles the device after any operation*
- [ATCA\\_STATUS\\_calib\\_cfg\\_discover](#) ([ATCAInterfaceCfg](#) cfg\_array[], int max)  
*auto discovery of crypto auth devices*
- [ATCA\\_STATUS\\_calib\\_get\\_addr](#) (uint8\_t zone, uint16\_t slot, uint8\_t block, uint8\_t offset, uint16\_t \*addr)  
*Compute the address given the zone, slot, block, and offset.*
- [ATCA\\_STATUS\\_calib\\_get\\_zone\\_size](#) ([ATCADevice](#) device, uint8\_t zone, uint16\_t slot, size\_t \*size)  
*Gets the size of the specified zone in bytes.*
- [ATCA\\_STATUS\\_calib\\_aes](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*aes\_in, uint8\_t \*aes\_out)  
*Compute the AES-128 encrypt, decrypt, or GFM calculation.*
- [ATCA\\_STATUS\\_calib\\_aes\\_encrypt](#) ([ATCADevice](#) device, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*plaintext, uint8\_t \*ciphertext)  
*Perform an AES-128 encrypt operation with a key in the device.*
- [ATCA\\_STATUS\\_calib\\_aes\\_decrypt](#) ([ATCADevice](#) device, uint16\_t key\_id, uint8\_t key\_block, const uint8\_t \*ciphertext, uint8\_t \*plaintext)  
*Perform an AES-128 decrypt operation with a key in the device.*
- [ATCA\\_STATUS\\_calib\\_aes\\_gfm](#) ([ATCADevice](#) device, const uint8\_t \*h, const uint8\_t \*input, uint8\_t \*output)  
*Perform a Galois Field Multiply (GFM) operation.*
- [ATCA\\_STATUS\\_calib\\_checkmac](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*challenge, const uint8\_t \*response, const uint8\_t \*other\_data)  
*Compares a MAC response with input values.*
- [ATCA\\_STATUS\\_calib\\_counter](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Compute the Counter functions.*
- [ATCA\\_STATUS\\_calib\\_counter\\_increment](#) ([ATCADevice](#) device, uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Increments one of the device's monotonic counters.*
- [ATCA\\_STATUS\\_calib\\_counter\\_read](#) ([ATCADevice](#) device, uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Read one of the device's monotonic counters.*
- [ATCA\\_STATUS\\_calib\\_derivekey](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*mac)  
*Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.*
- [ATCA\\_STATUS\\_calib\\_ecdh\\_base](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, uint8\_t \*out\_nonce)  
*Base function for generating premaster secret key using ECDH.*
- [ATCA\\_STATUS\\_calib\\_ecdh](#) ([ATCADevice](#) device, uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms)  
*ECDH command with a private key in a slot and the premaster secret is returned in the clear.*

- **ATCA\_STATUS calib\_ecdh\_enc** (ATCADevice device, uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*read\_key, uint16\_t read\_key\_id, const uint8\_t num\_in[(20)])  
ECDH command with a private key in a slot and the premaster secret is returned encrypted using the IO protection key.
- **ATCA\_STATUS calib\_ecdh\_ioenc** (ATCADevice device, uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*io\_key)  
ECDH command with a private key in TempKey and the premaster secret is returned in the clear.
- **ATCA\_STATUS calib\_ecdh\_tempkey** (ATCADevice device, const uint8\_t \*public\_key, uint8\_t \*pms)  
ECDH command with a private key in TempKey and the premaster secret is returned encrypted using the IO protection key.
- **ATCA\_STATUS calib\_ecdh\_tempkey\_ioenc** (ATCADevice device, const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*io\_key)  
ECDH command with a private key in TempKey and the premaster secret is returned encrypted using the IO protection key.
- **ATCA\_STATUS calib\_gendig** (ATCADevice device, uint8\_t zone, uint16\_t key\_id, const uint8\_t \*other\_data, uint8\_t other\_data\_size)  
Issues a GenDig command, which performs a SHA256 hash on the source data indicated by zone with the contents of TempKey. See the CryptoAuth datasheet for your chip to see what the values of zone correspond to.
- **ATCA\_STATUS calib\_genkey\_base** (ATCADevice device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*other\_data, uint8\_t \*public\_key)  
Issues GenKey command, which can generate a private key, compute a public key, and/or compute a digest of a public key.
- **ATCA\_STATUS calib\_genkey** (ATCADevice device, uint16\_t key\_id, uint8\_t \*public\_key)  
Issues GenKey command, which generates a new random private key in slot and returns the public key.
- **ATCA\_STATUS calib\_get\_pubkey** (ATCADevice device, uint16\_t key\_id, uint8\_t \*public\_key)  
Uses GenKey command to calculate the public key from an existing private key in a slot.
- **ATCA\_STATUS calib\_hmac** (ATCADevice device, uint8\_t mode, uint16\_t key\_id, uint8\_t \*digest)  
Issues a HMAC command, which computes an HMAC/SHA-256 digest of a key stored in the device, a challenge, and other information on the device.
- **ATCA\_STATUS calib\_info\_base** (ATCADevice device, uint8\_t mode, uint16\_t param2, uint8\_t \*out\_data)  
Issues an Info command, which return internal device information and can control GPIO and the persistent latch.
- **ATCA\_STATUS calib\_info** (ATCADevice device, uint8\_t \*revision)  
Use the Info command to get the device revision (DevRev).
- **ATCA\_STATUS calib\_info\_set\_latch** (ATCADevice device, bool state)  
Use the Info command to set the persistent latch state for an ATECC608 device.
- **ATCA\_STATUS calib\_info\_get\_latch** (ATCADevice device, bool \*state)  
Use the Info command to get the persistent latch current state for an ATECC608 device.
- **ATCA\_STATUS calib\_kdf** (ATCADevice device, uint8\_t mode, uint16\_t key\_id, const uint32\_t details, const uint8\_t \*message, uint8\_t \*out\_data, uint8\_t \*out\_nonce)  
Executes the KDF command, which derives a new key in PRF, AES, or HKDF modes.
- **ATCA\_STATUS calib\_lock** (ATCADevice device, uint8\_t mode, uint16\_t summary\_crc)  
The Lock command prevents future modifications of the Configuration and/or Data and OTP zones. If the device is so configured, then this command can be used to lock individual data slots. This command fails if the designated area is already locked.
- **ATCA\_STATUS calib\_lock\_config\_zone** (ATCADevice device)  
Unconditionally (no CRC required) lock the config zone.
- **ATCA\_STATUS calib\_lock\_config\_zone\_crc** (ATCADevice device, uint16\_t summary\_crc)  
Lock the config zone with summary CRC.
- **ATCA\_STATUS calib\_lock\_data\_zone** (ATCADevice device)  
Unconditionally (no CRC required) lock the data zone (slots and OTP).
- **ATCA\_STATUS calib\_lock\_data\_zone\_crc** (ATCADevice device, uint16\_t summary\_crc)  
Lock the data zone (slots and OTP) with summary CRC.
- **ATCA\_STATUS calib\_lock\_data\_slot** (ATCADevice device, uint16\_t slot)  
Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (KeyConfig.Lockable=1).

- **ATCA\_STATUS calib\_mac** (ATCADevice device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*challenge, uint8\_t \*digest)  
*Executes MAC command, which computes a SHA-256 digest of a key stored in the device, a challenge, and other information on the device.*
- **ATCA\_STATUS calib\_nonce\_base** (ATCADevice device, uint8\_t mode, uint16\_t zero, const uint8\_t \*num\_in, uint8\_t \*rand\_out)  
*Executes Nonce command, which loads a random or fixed nonce/data into the device for use by subsequent commands.*
- **ATCA\_STATUS calib\_nonce** (ATCADevice device, const uint8\_t \*num\_in)  
*Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.*
- **ATCA\_STATUS calib\_nonce\_load** (ATCADevice device, uint8\_t target, const uint8\_t \*num\_in, uint16\_t num\_in\_size)  
*Execute a Nonce command in pass-through mode to load one of the device's internal buffers with a fixed value.*
- **ATCA\_STATUS calib\_nonce\_rand** (ATCADevice device, const uint8\_t \*num\_in, uint8\_t \*rand\_out)  
*Execute a Nonce command to generate a random nonce combining a host nonce (num\_in) and a device random number.*
- **ATCA\_STATUS calib\_challenge** (ATCADevice device, const uint8\_t \*num\_in)  
*Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.*
- **ATCA\_STATUS calib\_challenge\_seed\_update** (ATCADevice device, const uint8\_t \*num\_in, uint8\_t \*rand\_out)  
*Execute a Nonce command to generate a random challenge combining a host nonce (num\_in) and a device random number.*
- **ATCA\_STATUS calib\_priv\_write** (ATCADevice device, uint16\_t key\_id, const uint8\_t priv\_key[36], uint16\_t write\_key\_id, const uint8\_t write\_key[32], const uint8\_t num\_in[(20)])
- **ATCA\_STATUS calib\_random** (ATCADevice device, uint8\_t \*rand\_out)  
*Executes Random command, which generates a 32 byte random number from the CryptoAuth device.*
- **ATCA\_STATUS calib\_read\_zone** (ATCADevice device, uint8\_t zone, uint16\_t slot, uint8\_t block, uint8\_t offset, uint8\_t \*data, uint8\_t len)  
*Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.*
- **ATCA\_STATUS calib\_is\_locked** (ATCADevice device, uint8\_t zone, bool \*is\_locked)  
*Executes Read command, which reads the configuration zone to see if the specified zone is locked.*
- **ATCA\_STATUS calib\_is\_slot\_locked** (ATCADevice device, uint16\_t slot, bool \*is\_locked)  
*Executes Read command, which reads the configuration zone to see if the specified slot is locked.*
- **ATCA\_STATUS calib\_read\_bytes\_zone** (ATCADevice device, uint8\_t zone, uint16\_t slot, size\_t offset, uint8\_t \*data, size\_t length)  
*Used to read an arbitrary number of bytes from any zone configured for clear reads.*
- **ATCA\_STATUS calib\_read\_serial\_number** (ATCADevice device, uint8\_t \*serial\_number)  
*Executes Read command, which reads the 9 byte serial number of the device from the config zone.*
- **ATCA\_STATUS calib\_read\_pubkey** (ATCADevice device, uint16\_t slot, uint8\_t \*public\_key)  
*Executes Read command to read an ECC P256 public key from a slot configured for clear reads.*
- **ATCA\_STATUS calib\_read\_sig** (ATCADevice device, uint16\_t slot, uint8\_t \*sig)  
*Executes Read command to read a 64 byte ECDSA P256 signature from a slot configured for clear reads.*
- **ATCA\_STATUS calib\_read\_config\_zone** (ATCADevice device, uint8\_t \*config\_data)  
*Executes Read command to read the complete device configuration zone.*
- **ATCA\_STATUS calib\_cmp\_config\_zone** (ATCADevice device, uint8\_t \*config\_data, bool \*same\_config)  
*Compares a specified configuration zone with the configuration zone currently on the device.*
- **ATCA\_STATUS calib\_read\_enc** (ATCADevice device, uint16\_t key\_id, uint8\_t block, uint8\_t \*data, const uint8\_t \*enc\_key, const uint16\_t enc\_key\_id, const uint8\_t num\_in[(20)])
- **ATCA\_STATUS calib\_secureboot** (ATCADevice device, uint8\_t mode, uint16\_t param2, const uint8\_t \*digest, const uint8\_t \*signature, uint8\_t \*mac)  
*Executes Secure Boot command, which provides support for secure boot of an external MCU or MPU.*

- **ATCA\_STATUS calib\_secureboot\_mac** (ATCADevice device, uint8\_t mode, const uint8\_t \*digest, const uint8\_t \*signature, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)  
*Executes Secure Boot command with encrypted digest and validated MAC response using the IO protection key.*
- **ATCA\_STATUS calib\_selftest** (ATCADevice device, uint8\_t mode, uint16\_t param2, uint8\_t \*result)  
*Executes the SelfTest command, which performs a test of one or more of the cryptographic engines within the AT↵ ECC608 chip.*
- **ATCA\_STATUS calib\_sha\_base** (ATCADevice device, uint8\_t mode, uint16\_t length, const uint8\_t \*data\_in, uint8\_t \*data\_out, uint16\_t \*data\_out\_size)  
*Executes SHA command, which computes a SHA-256 or HMAC/SHA-256 digest for general purpose use by the host system.*
- **ATCA\_STATUS calib\_sha\_start** (ATCADevice device)  
*Executes SHA command to initialize SHA-256 calculation engine.*
- **ATCA\_STATUS calib\_sha\_update** (ATCADevice device, const uint8\_t \*message)  
*Executes SHA command to add 64 bytes of message data to the current context.*
- **ATCA\_STATUS calib\_sha\_end** (ATCADevice device, uint8\_t \*digest, uint16\_t length, const uint8\_t ↵ \*message)  
*Executes SHA command to complete SHA-256 or HMAC/SHA-256 operation.*
- **ATCA\_STATUS calib\_sha\_read\_context** (ATCADevice device, uint8\_t \*context, uint16\_t \*context\_size)  
*Executes SHA command to read the SHA-256 context back. Only for ATECC608 with SHA-256 contexts. HMAC not supported.*
- **ATCA\_STATUS calib\_sha\_write\_context** (ATCADevice device, const uint8\_t \*context, uint16\_t context\_size)  
*Executes SHA command to write (restore) a SHA-256 context into the the device. Only supported for ATECC608 with SHA-256 contexts.*
- **ATCA\_STATUS calib\_sha** (ATCADevice device, uint16\_t length, const uint8\_t \*message, uint8\_t \*digest)  
*Use the SHA command to compute a SHA-256 digest.*
- **ATCA\_STATUS calib\_hw\_sha2\_256** (ATCADevice device, const uint8\_t \*data, size\_t data\_size, uint8\_t ↵ \*digest)  
*Use the SHA command to compute a SHA-256 digest.*
- **ATCA\_STATUS calib\_hw\_sha2\_256\_init** (ATCADevice device, atca\_sha256\_ctx\_t \*ctx)  
*Initialize a SHA context for performing a hardware SHA-256 operation on a device. Note that only one SHA operation can be run at a time.*
- **ATCA\_STATUS calib\_hw\_sha2\_256\_update** (ATCADevice device, atca\_sha256\_ctx\_t \*ctx, const uint8\_t ↵ \*data, size\_t data\_size)  
*Add message data to a SHA context for performing a hardware SHA-256 operation on a device.*
- **ATCA\_STATUS calib\_hw\_sha2\_256\_finish** (ATCADevice device, atca\_sha256\_ctx\_t \*ctx, uint8\_t \*digest)  
*Finish SHA-256 digest for a SHA context for performing a hardware SHA-256 operation on a device.*
- **ATCA\_STATUS calib\_sha\_hmac\_init** (ATCADevice device, atca\_hmac\_sha256\_ctx\_t \*ctx, uint16\_t key ↵ slot)  
*Executes SHA command to start an HMAC/SHA-256 operation.*
- **ATCA\_STATUS calib\_sha\_hmac\_update** (ATCADevice device, atca\_hmac\_sha256\_ctx\_t \*ctx, const uint8\_t ↵ \*data, size\_t data\_size)  
*Executes SHA command to add an arbitrary amount of message data to a HMAC/SHA-256 operation.*
- **ATCA\_STATUS calib\_sha\_hmac\_finish** (ATCADevice device, atca\_hmac\_sha256\_ctx\_t \*ctx, uint8\_t ↵ \*digest, uint8\_t target)  
*Executes SHA command to complete a HMAC/SHA-256 operation.*
- **ATCA\_STATUS calib\_sha\_hmac** (ATCADevice device, const uint8\_t \*data, size\_t data\_size, uint16\_t key ↵ slot, uint8\_t \*digest, uint8\_t target)  
*Use the SHA command to compute an HMAC/SHA-256 operation.*
- **ATCA\_STATUS calib\_sign\_base** (ATCADevice device, uint8\_t mode, uint16\_t key\_id, uint8\_t \*signature)  
*Executes the Sign command, which generates a signature using the ECDSA algorithm.*
- **ATCA\_STATUS calib\_sign** (ATCADevice device, uint16\_t key\_id, const uint8\_t \*msg, uint8\_t \*signature)  
*Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*



- **ATCA\_STATUS calib\_sign\_internal** (ATCADevice device, uint16\_t key\_id, bool is\_invalidate, bool is\_full\_sn, uint8\_t \*signature)  
*Executes Sign command to sign an internally generated message.*
- **ATCA\_STATUS calib\_updateextra** (ATCADevice device, uint8\_t mode, uint16\_t new\_value)  
*Executes UpdateExtra command to update the values of the two extra bytes within the Configuration zone (bytes 84 and 85).*
- **ATCA\_STATUS calib\_verify** (ATCADevice device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*public\_key, const uint8\_t \*other\_data, uint8\_t \*mac)  
*Executes the Verify command, which takes an ECDSA [R,S] signature and verifies that it is correctly generated from a given message and public key. In all cases, the signature is an input to the command.*
- **ATCA\_STATUS calib\_verify\_extern** (ATCADevice device, const uint8\_t \*message, const uint8\_t \*signature, const uint8\_t \*public\_key, bool \*is\_verified)  
*Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*
- **ATCA\_STATUS calib\_verify\_extern\_mac** (ATCADevice device, const uint8\_t \*message, const uint8\_t \*signature, const uint8\_t \*public\_key, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)  
*Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. This function is only available on the ATECC608.*
- **ATCA\_STATUS calib\_verify\_stored** (ATCADevice device, const uint8\_t \*message, const uint8\_t \*signature, uint16\_t key\_id, bool \*is\_verified)  
*Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*
- **ATCA\_STATUS calib\_verify\_stored\_mac** (ATCADevice device, const uint8\_t \*message, const uint8\_t \*signature, uint16\_t key\_id, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)  
*Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with a public key stored in the device. This function is only available on the ATECC608.*
- **ATCA\_STATUS calib\_verify\_validate** (ATCADevice device, uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*other\_data, bool \*is\_verified)  
*Executes the Verify command in Validate mode to validate a public key stored in a slot.*
- **ATCA\_STATUS calib\_verify\_invalidate** (ATCADevice device, uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*other\_data, bool \*is\_verified)  
*Executes the Verify command in Invalidate mode which invalidates a previously validated public key stored in a slot.*
- **ATCA\_STATUS calib\_write** (ATCADevice device, uint8\_t zone, uint16\_t address, const uint8\_t \*value, const uint8\_t \*mac)  
*Executes the Write command, which writes either one four byte word or a 32-byte block to one of the EEPROM zones on the device. Depending upon the value of the WriteConfig byte for this slot, the data may be required to be encrypted by the system prior to being sent to the device. This command cannot be used to write slots configured as ECC private keys.*
- **ATCA\_STATUS calib\_write\_zone** (ATCADevice device, uint8\_t zone, uint16\_t slot, uint8\_t block, uint8\_t offset, const uint8\_t \*data, uint8\_t len)  
*Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.*
- **ATCA\_STATUS calib\_write\_bytes\_zone** (ATCADevice device, uint8\_t zone, uint16\_t slot, size\_t offset\_bytes, const uint8\_t \*data, size\_t length)  
*Executes the Write command, which writes data into the configuration, otp, or data zones with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).*
- **ATCA\_STATUS calib\_write\_pubkey** (ATCADevice device, uint16\_t slot, const uint8\_t \*public\_key)  
*Uses the write command to write a public key to a slot in the proper format.*
- **ATCA\_STATUS calib\_write\_config\_zone** (ATCADevice device, const uint8\_t \*config\_data)  
*Executes the Write command, which writes the configuration zone.*
- **ATCA\_STATUS calib\_write\_enc** (ATCADevice device, uint16\_t key\_id, uint8\_t block, const uint8\_t \*data, const uint8\_t \*enc\_key, const uint16\_t enc\_key\_id, const uint8\_t num\_in[(20)])
- **ATCA\_STATUS calib\_write\_config\_counter** (ATCADevice device, uint16\_t counter\_id, uint32\_t counter\_value)  
*Initialize one of the monotonic counters in device with a specific value.*

## 20.69 calib\_checkmac.c File Reference

CryptoAuthLib Basic API methods for CheckMAC command.

```
#include "cryptoauthlib.h"
```

### Functions

- [ATCA\\_STATUS](#) [calib\\_checkmac](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*challenge, const uint8\_t \*response, const uint8\_t \*other\_data)

*Compares a MAC response with input values.*

### 20.69.1 Detailed Description

CryptoAuthLib Basic API methods for CheckMAC command.

The CheckMac command calculates a MAC response that would have been generated on a different CryptoAuth Authentication device and then compares the result with input value.

#### Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.70 calib\_command.c File Reference

Microchip CryptoAuthentication device command builder - this is the main object that builds the command byte strings for the given device. It does not execute the command. The basic flow is to call a command method to build the command you want given the parameters and then send that byte string through the device interface.

```
#include "cryptoauthlib.h"
```



## Functions

- [ATCA\\_STATUS atCheckMAC](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand CheckMAC method.*
- [ATCA\\_STATUS atCounter](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand Counter method.*
- [ATCA\\_STATUS atDeriveKey](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet, bool has\_mac)  
*ATCACommand DeriveKey method.*
- [ATCA\\_STATUS atECDH](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand ECDH method.*
- [ATCA\\_STATUS atGenDig](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet, bool is\_no\_mac\_key)  
*ATCACommand Generate Digest method.*
- [ATCA\\_STATUS atGenKey](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand Generate Key method.*
- [ATCA\\_STATUS atHMAC](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand HMAC method.*
- [ATCA\\_STATUS atInfo](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand Info method.*
- [ATCA\\_STATUS atLock](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand Lock method.*
- [ATCA\\_STATUS atMAC](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand MAC method.*
- [ATCA\\_STATUS atNonce](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand Nonce method.*
- [ATCA\\_STATUS atPause](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand Pause method.*
- [ATCA\\_STATUS atPrivWrite](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand PrivWrite method.*
- [ATCA\\_STATUS atRandom](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand Random method.*
- [ATCA\\_STATUS atRead](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand Read method.*
- [ATCA\\_STATUS atSecureBoot](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand SecureBoot method.*
- [ATCA\\_STATUS atSHA](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet, uint16\_t write\_context\_size)  
*ATCACommand SHA method.*
- [ATCA\\_STATUS atSign](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand Sign method.*
- [ATCA\\_STATUS atUpdateExtra](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand UpdateExtra method.*
- [ATCA\\_STATUS atVerify](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand ECDSA Verify method.*
- [ATCA\\_STATUS atWrite](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet, bool has\_mac)  
*ATCACommand Write method.*
- [ATCA\\_STATUS atAES](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand AES method.*
- [ATCA\\_STATUS atSelfTest](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand AES method.*
- [ATCA\\_STATUS atKDF](#) ([ATCACommand](#) ca\_cmd, [ATCAPacket](#) \*packet)  
*ATCACommand KDF method.*
- void [atCRC](#) (size\_t length, const uint8\_t \*data, uint8\_t \*crc\_le)

*Calculates CRC over the given raw data and returns the CRC in little-endian byte order.*

- void [atCalcCrc](#) ([ATCAPacket](#) \*packet)

*This function calculates CRC and adds it to the correct offset in the packet data.*

- [ATCA\\_STATUS](#) [atCheckCrc](#) (const uint8\_t \*response)

*This function checks the consistency of a response.*

- bool [atIsSHAFamily](#) ([ATCADeviceType](#) device\_type)

*determines if a given device type is a SHA device or a superset of a SHA device*

- bool [atIsECCFamily](#) ([ATCADeviceType](#) device\_type)

*determines if a given device type is an ECC device or a superset of a ECC device*

- [ATCA\\_STATUS](#) [isATCAError](#) (uint8\_t \*data)

*checks for basic error frame in data*

## 20.70.1 Detailed Description

Microchip CryptoAuthentication device command builder - this is the main object that builds the command byte strings for the given device. It does not execute the command. The basic flow is to call a command method to build the command you want given the parameters and then send that byte string through the device interface.

The primary goal of the command builder is to wrap the given parameters with the correct packet size and CRC. The caller should first fill in the parameters required in the [ATCAPacket](#) parameter given to the command. The command builder will deal with the mechanics of creating a valid packet using the parameter information.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.70.2 Function Documentation

### 20.70.2.1 atAES()

```
ATCA\_STATUS atAES (  
    ATCACommand ca_cmd,  
    ATCAPacket * packet )
```

ATCACommand AES method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

#### Returns

[ATCA\\_SUCCESS](#)

### 20.70.2.2 atCalcCrc()

```
void atCalcCrc (
    ATCAPacket * packet )
```

This function calculates CRC and adds it to the correct offset in the packet data.

#### Parameters

in	<i>packet</i>	Packet to calculate CRC data for
----	---------------	----------------------------------

### 20.70.2.3 atCheckCrc()

```
ATCA_STATUS atCheckCrc (
    const uint8_t * response )
```

This function checks the consistency of a response.

#### Parameters

in	<i>response</i>	pointer to response
----	-----------------	---------------------

#### Returns

ATCA\_SUCCESS on success, otherwise ATCA\_RX\_CRC\_ERROR

### 20.70.2.4 atCheckMAC()

```
ATCA_STATUS atCheckMAC (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand CheckMAC method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

#### Returns

ATCA\_SUCCESS

### 20.70.2.5 atCounter()

```
ATCA_STATUS atCounter (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Counter method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.70.2.6 atCRC()

```
void atCRC (
    size_t length,
    const uint8_t * data,
    uint8_t * crc_le )
```

Calculates CRC over the given raw data and returns the CRC in little-endian byte order.

#### Parameters

in	<i>length</i>	Size of data not including the CRC byte positions
in	<i>data</i>	Pointer to the data over which to compute the CRC
out	<i>crc↔ _le</i>	Pointer to the place where the two-bytes of CRC will be returned in little-endian byte order.

### 20.70.2.7 atDeriveKey()

```
ATCA_STATUS atDeriveKey (
    ATCACommand ca_cmd,
    ATCAPacket * packet,
    bool has_mac )
```

ATCACommand DeriveKey method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built
in	<i>has_mac</i>	hasMAC determines if MAC data is present in the packet input

**Returns**

ATCA\_SUCCESS

**20.70.2.8 atECDH()**

```
ATCA_STATUS atECDH (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand ECDH method.

**Parameters**

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

**Returns**

ATCA\_SUCCESS

**20.70.2.9 atGenDig()**

```
ATCA_STATUS atGenDig (
    ATCACommand ca_cmd,
    ATCAPacket * packet,
    bool is_no_mac_key )
```

ATCACommand Generate Digest method.

**Parameters**

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built
in	<i>is_no_mac_key</i>	Should be true if GenDig is being run on a slot that has its SlotConfig.NoMac bit set

**Returns**

ATCA\_SUCCESS

**20.70.2.10 atGenKey()**

```
ATCA_STATUS atGenKey (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Generate Key method.

### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

### Returns

ATCA\_SUCCESS

## 20.70.2.11 atHMAC()

```
ATCA_STATUS atHMAC (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand HMAC method.

### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

### Returns

ATCA\_SUCCESS

## 20.70.2.12 atInfo()

```
ATCA_STATUS atInfo (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Info method.

### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

### Returns

ATCA\_SUCCESS

### 20.70.2.13 atIsECCFamily()

```
bool atIsECCFamily (
    ATCADeviceType device_type )
```

determines if a given device type is an ECC device or a superset of a ECC device

#### Parameters

in	<i>device_type</i>	Type of device to check for family type
----	--------------------	---

#### Returns

boolean indicating whether the given device is an ECC family device.

### 20.70.2.14 atIsSHAFamily()

```
bool atIsSHAFamily (
    ATCADeviceType device_type )
```

determines if a given device type is a SHA device or a superset of a SHA device

#### Parameters

in	<i>device_type</i>	Type of device to check for family type
----	--------------------	---

#### Returns

boolean indicating whether the given device is a SHA family device.

### 20.70.2.15 atKDF()

```
ATCA_STATUS atKDF (
    ATCACCommand ca_cmd,
    ATCAPacket * packet )
```

ATCACCommand KDF method.

#### Parameters

in	<i>ca_cmd</i>	Instance
in	<i>packet</i>	Pointer to the packet containing the command being built.

### Returns

ATCA\_SUCCESS

### 20.70.2.16 atLock()

```
ATCA_STATUS atLock (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Lock method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

### Returns

ATCA\_SUCCESS

### 20.70.2.17 atMAC()

```
ATCA_STATUS atMAC (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand MAC method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

### Returns

ATCA\_SUCCESS

### 20.70.2.18 atNonce()

```
ATCA_STATUS atNonce (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Nonce method.



**Parameters**

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.70.2.19 atPause()**

```
ATCA_STATUS atPause (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Pause method.

**Parameters**

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

**Returns**

ATCA\_SUCCESS

**20.70.2.20 atPrivWrite()**

```
ATCA_STATUS atPrivWrite (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand PrivWrite method.

**Parameters**

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

**Returns**

ATCA\_SUCCESS

**20.70.2.21 atRandom()**

```
ATCA_STATUS atRandom (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Random method.

**Parameters**

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

**Returns**

ATCA\_SUCCESS

**20.70.2.22 atRead()**

```
ATCA_STATUS atRead (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Read method.

**Parameters**

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

**Returns**

ATCA\_SUCCESS

**20.70.2.23 atSecureBoot()**

```
ATCA_STATUS atSecureBoot (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand SecureBoot method.

**Parameters**

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

**Returns**

ATCA\_SUCCESS

**20.70.2.24 atSelfTest()**

```
ATCA_STATUS atSelfTest (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand AES method.

**Parameters**

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

**Returns**

ATCA\_SUCCESS

**20.70.2.25 atSHA()**

```
ATCA_STATUS atSHA (
    ATCACommand ca_cmd,
    ATCAPacket * packet,
    uint16_t write_context_size )
```

ATCACommand SHA method.

**Parameters**

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built
in	<i>write_context_size</i>	the length of the sha write_context data

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.70.2.26 atSign()**

```
ATCA_STATUS atSign (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Sign method.

### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

### Returns

ATCA\_SUCCESS

### 20.70.2.27 atUpdateExtra()

```
ATCA_STATUS atUpdateExtra (  
    ATCACommand ca_cmd,  
    ATCAPacket * packet )
```

ATCACommand UpdateExtra method.

### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

### Returns

ATCA\_SUCCESS

### 20.70.2.28 atVerify()

```
ATCA_STATUS atVerify (  
    ATCACommand ca_cmd,  
    ATCAPacket * packet )
```

ATCACommand ECDSA Verify method.

### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

**20.70.2.29 atWrite()**

```
ATCA_STATUS atWrite (
    ATCACommand ca_cmd,
    ATCAPacket * packet,
    bool has_mac )
```

ATCACommand Write method.

**Parameters**

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built
in	<i>has_mac</i>	Flag to indicate whether a mac is present or not

**Returns**

ATCA\_SUCCESS

**20.70.2.30 isATCAError()**

```
ATCA_STATUS isATCAError (
    uint8_t * data )
```

checks for basic error frame in data

**Parameters**

in	<i>data</i>	pointer to received data - expected to be in the form of a CA device response frame
----	-------------	---

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.71 calib\_command.h File Reference**

Microchip Crypto Auth device command object - this is a command builder only, it does not send the command. The result of a command method is a fully formed packet, ready to send to the ATCAIFace object to dispatch.

```
#include <stddef.h>
#include "atca_command.h"
```

## Data Structures

- struct [ATCAPacket](#)

## Macros

- #define [ATCA\\_CMD\\_SIZE\\_MIN](#) ((uint8\_t)7)  
*minimum number of bytes in command (from count byte to second CRC byte)*
- #define [ATCA\\_CMD\\_SIZE\\_MAX](#) ((uint8\_t)4 \* 36 + 7)  
*maximum size of command packet (Verify)*
- #define [CMD\\_STATUS\\_SUCCESS](#) ((uint8\_t)0x00)  
*status byte for success*
- #define [CMD\\_STATUS\\_WAKEUP](#) ((uint8\_t)0x11)  
*status byte after wake-up*
- #define [CMD\\_STATUS\\_BYTE\\_PARSE](#) ((uint8\_t)0x03)  
*command parse error*
- #define [CMD\\_STATUS\\_BYTE\\_ECC](#) ((uint8\_t)0x05)  
*command ECC error*
- #define [CMD\\_STATUS\\_BYTE\\_EXEC](#) ((uint8\_t)0x0F)  
*command execution error*
- #define [CMD\\_STATUS\\_BYTE\\_COMM](#) ((uint8\_t)0xFF)  
*communication error*

## Opcodes for Crypto Authentication device commands

- #define [ATCA\\_CHECKMAC](#) ((uint8\_t)0x28)  
*CheckMac command op-code.*
- #define [ATCA\\_DERIVE\\_KEY](#) ((uint8\_t)0x1C)  
*DeriveKey command op-code.*
- #define [ATCA\\_INFO](#) ((uint8\_t)0x30)  
*Info command op-code.*
- #define [ATCA\\_GENDIG](#) ((uint8\_t)0x15)  
*GenDig command op-code.*
- #define [ATCA\\_GENKEY](#) ((uint8\_t)0x40)  
*GenKey command op-code.*
- #define [ATCA\\_HMAC](#) ((uint8\_t)0x11)  
*HMAC command op-code.*
- #define [ATCA\\_LOCK](#) ((uint8\_t)0x17)  
*Lock command op-code.*
- #define [ATCA\\_MAC](#) ((uint8\_t)0x08)  
*MAC command op-code.*
- #define [ATCA\\_NONCE](#) ((uint8\_t)0x16)  
*Nonce command op-code.*
- #define [ATCA\\_PAUSE](#) ((uint8\_t)0x01)  
*Pause command op-code.*
- #define [ATCA\\_PRIVWRITE](#) ((uint8\_t)0x46)  
*PrivWrite command op-code.*
- #define [ATCA\\_RANDOM](#) ((uint8\_t)0x1B)  
*Random command op-code.*
- #define [ATCA\\_READ](#) ((uint8\_t)0x02)  
*Read command op-code.*
- #define [ATCA\\_SIGN](#) ((uint8\_t)0x41)  
*Sign command op-code.*
- #define [ATCA\\_UPDATE\\_EXTRA](#) ((uint8\_t)0x20)

- *UpdateExtra command op-code.*
- #define [ATCA\\_VERIFY](#) ((uint8\_t)0x45)
- *GenKey command op-code.*
- #define [ATCA\\_WRITE](#) ((uint8\_t)0x12)
- *Write command op-code.*
- #define [ATCA\\_ECDH](#) ((uint8\_t)0x43)
- *ECDH command op-code.*
- #define [ATCA\\_COUNTER](#) ((uint8\_t)0x24)
- *Counter command op-code.*
- #define [ATCA\\_SHA](#) ((uint8\_t)0x47)
- *SHA command op-code.*
- #define [ATCA\\_AES](#) ((uint8\_t)0x51)
- *AES command op-code.*
- #define [ATCA\\_KDF](#) ((uint8\_t)0x56)
- *KDF command op-code.*
- #define [ATCA\\_SECUREBOOT](#) ((uint8\_t)0x80)
- *Secure Boot command op-code.*
- #define [ATCA\\_SELFTEST](#) ((uint8\_t)0x77)
- *Self test command op-code.*

## Definitions of Data and Packet Sizes

- #define [ATCA\\_BLOCK\\_SIZE](#) (32)
- *size of a block*
- #define [ATCA\\_WORD\\_SIZE](#) (4)
- *size of a word*
- #define [ATCA\\_PUB\\_KEY\\_PAD](#) (4)
- *size of the public key pad*
- #define [ATCA\\_SERIAL\\_NUM\\_SIZE](#) (9)
- *number of bytes in the device serial number*
- #define [ATCA\\_RSP\\_SIZE\\_VAL](#) ((uint8\_t)7)
- *size of response packet containing four bytes of data*
- #define [ATCA\\_KEY\\_COUNT](#) (16)
- *number of keys*
- #define [ATCA\\_ECC\\_CONFIG\\_SIZE](#) (128)
- *size of configuration zone*
- #define [ATCA\\_SHA\\_CONFIG\\_SIZE](#) (88)
- *size of configuration zone*
- #define [ATCA\\_OTP\\_SIZE](#) (64)
- *size of OTP zone*
- #define [ATCA\\_DATA\\_SIZE](#) (ATCA\_KEY\_COUNT \* ATCA\_KEY\_SIZE)
- *size of data zone*
- #define [ATCA\\_AES\\_GFM\\_SIZE](#) ATCA\_BLOCK\_SIZE
- *size of GFM data*
- #define [ATCA\\_CHIPMODE\\_OFFSET](#) (19)
- *ChipMode byte offset within the configuration zone.*
- #define [ATCA\\_CHIPMODE\\_I2C\\_ADDRESS\\_FLAG](#) ((uint8\_t)0x01)
- *ChipMode I2C Address in UserExtraAdd flag.*
- #define [ATCA\\_CHIPMODE\\_TTL\\_ENABLE\\_FLAG](#) ((uint8\_t)0x02)
- *ChipMode TTLEnable flag.*
- #define [ATCA\\_CHIPMODE\\_WATCHDOG\\_MASK](#) ((uint8\_t)0x04)
- *ChipMode watchdog duration mask.*
- #define [ATCA\\_CHIPMODE\\_WATCHDOG\\_SHORT](#) ((uint8\_t)0x00)
- *ChipMode short watchdog (~1.3s)*
- #define [ATCA\\_CHIPMODE\\_WATCHDOG\\_LONG](#) ((uint8\_t)0x04)
- *ChipMode long watchdog (~13s)*
- #define [ATCA\\_CHIPMODE\\_CLOCK\\_DIV\\_MASK](#) ((uint8\_t)0xF8)
- *ChipMode clock divider mask.*

- #define [ATCA\\_CHIPMODE\\_CLOCK\\_DIV\\_M0](#) ((uint8\_t)0x00)  
*ChipMode clock divider M0.*
- #define [ATCA\\_CHIPMODE\\_CLOCK\\_DIV\\_M1](#) ((uint8\_t)0x28)  
*ChipMode clock divider M1.*
- #define [ATCA\\_CHIPMODE\\_CLOCK\\_DIV\\_M2](#) ((uint8\_t)0x68)  
*ChipMode clock divider M2.*
- #define [ATCA\\_COUNT\\_SIZE](#) ((uint8\_t)1)  
*Number of bytes in the command packet Count.*
- #define [ATCA\\_CRC\\_SIZE](#) ((uint8\_t)2)  
*Number of bytes in the command packet CRC.*
- #define [ATCA\\_PACKET\\_OVERHEAD](#) (ATCA\_COUNT\_SIZE + ATCA\_CRC\_SIZE)  
*Number of bytes in the command packet.*
- #define [ATCA\\_PUB\\_KEY\\_SIZE](#) (64)  
*size of a p256 public key*
- #define [ATCA\\_PRIV\\_KEY\\_SIZE](#) (32)  
*size of a p256 private key*
- #define [ATCA\\_SIG\\_SIZE](#) (64)  
*size of a p256 signature*
- #define [ATCA\\_KEY\\_SIZE](#) (32)  
*size of a symmetric SHA key*
- #define [RSA2048\\_KEY\\_SIZE](#) (256)  
*size of a RSA private key*
- #define [ATCA\\_RSP\\_SIZE\\_MIN](#) ((uint8\_t)4)  
*minimum number of bytes in response*
- #define [ATCA\\_RSP\\_SIZE\\_4](#) ((uint8\_t)7)  
*size of response packet containing 4 bytes data*
- #define [ATCA\\_RSP\\_SIZE\\_72](#) ((uint8\_t)75)  
*size of response packet containing 64 bytes data*
- #define [ATCA\\_RSP\\_SIZE\\_64](#) ((uint8\_t)67)  
*size of response packet containing 64 bytes data*
- #define [ATCA\\_RSP\\_SIZE\\_32](#) ((uint8\_t)35)  
*size of response packet containing 32 bytes data*
- #define [ATCA\\_RSP\\_SIZE\\_16](#) ((uint8\_t)19)  
*size of response packet containing 16 bytes data*
- #define [ATCA\\_RSP\\_SIZE\\_MAX](#) ((uint8\_t)75)  
*maximum size of response packet (GenKey and Verify command)*
- #define [OUTNONCE\\_SIZE](#) (32)  
*Size of the OutNonce response expected from several commands.*

### Definitions for Command Parameter Ranges

- #define [ATCA\\_KEY\\_ID\\_MAX](#) ((uint8\_t)15)  
*maximum value for key id*
- #define [ATCA\\_OTP\\_BLOCK\\_MAX](#) ((uint8\_t)1)  
*maximum value for OTP block*

### Definitions for Indexes Common to All Commands

- #define [ATCA\\_COUNT\\_IDX](#) (0)  
*command packet index for count*
- #define [ATCA\\_OPCODE\\_IDX](#) (1)  
*command packet index for op-code*
- #define [ATCA\\_PARAM1\\_IDX](#) (2)  
*command packet index for first parameter*
- #define [ATCA\\_PARAM2\\_IDX](#) (3)  
*command packet index for second parameter*
- #define [ATCA\\_DATA\\_IDX](#) (5)  
*command packet index for data load*



- #define `ATCA_RSP_DATA_IDX` (1)  
*buffer index of data in response*

### Definitions for Zone and Address Parameters

- #define `ATCA_ZONE_MASK` ((uint8\_t)0x03)  
*Zone mask.*
- #define `ATCA_ZONE_ENCRYPTED` ((uint8\_t)0x40)  
*Zone bit 6 set: Write is encrypted with an unlocked data zone.*
- #define `ATCA_ZONE_READWRITE_32` ((uint8\_t)0x80)  
*Zone bit 7 set: Access 32 bytes, otherwise 4 bytes.*
- #define `ATCA_ADDRESS_MASK_CONFIG` (0x001F)  
*Address bits 5 to 7 are 0 for Configuration zone.*
- #define `ATCA_ADDRESS_MASK_OTP` (0x000F)  
*Address bits 4 to 7 are 0 for OTP zone.*
- #define `ATCA_ADDRESS_MASK` (0x007F)  
*Address bit 7 to 15 are always 0.*
- #define `ATCA_TEMPKEY_KEYID` (0xFFFF)  
*KeyID when referencing TempKey.*

### Definitions for Key types

- #define `ATCA_B283_KEY_TYPE` 0  
*B283 NIST ECC key.*
- #define `ATCA_K283_KEY_TYPE` 1  
*K283 NIST ECC key.*
- #define `ATCA_P256_KEY_TYPE` 4  
*P256 NIST ECC key.*
- #define `ATCA_AES_KEY_TYPE` 6  
*AES-128 Key.*
- #define `ATCA_SHA_KEY_TYPE` 7  
*SHA key or other data.*

### Definitions for the AES Command

- #define `AES_MODE_IDX ATCA_PARAM1_IDX`  
*AES command index for mode.*
- #define `AES_KEYID_IDX ATCA_PARAM2_IDX`  
*AES command index for key id.*
- #define `AES_INPUT_IDX ATCA_DATA_IDX`  
*AES command index for input data.*
- #define `AES_COUNT` (23)  
*AES command packet size.*
- #define `AES_MODE_MASK` ((uint8\_t)0xC7)  
*AES mode bits 3 to 5 are 0.*
- #define `AES_MODE_KEY_BLOCK_MASK` ((uint8\_t)0xC0)  
*AES mode mask for key block field.*
- #define `AES_MODE_OP_MASK` ((uint8\_t)0x07)  
*AES mode operation mask.*
- #define `AES_MODE_ENCRYPT` ((uint8\_t)0x00)  
*AES mode: Encrypt.*
- #define `AES_MODE_DECRYPT` ((uint8\_t)0x01)  
*AES mode: Decrypt.*
- #define `AES_MODE_GFM` ((uint8\_t)0x03)  
*AES mode: GFM calculation.*
- #define `AES_MODE_KEY_BLOCK_POS` (6)  
*Bit shift for key block in mode.*

- #define `AES_DATA_SIZE` (16)  
*size of AES encrypt/decrypt data*
- #define `AES_RSP_SIZE ATCA_RSP_SIZE_16`  
*AES command response packet size.*

#### Definitions for the CheckMac Command

- #define `CHECKMAC_MODE_IDX ATCA_PARAM1_IDX`  
*CheckMAC command index for mode.*
- #define `CHECKMAC_KEYID_IDX ATCA_PARAM2_IDX`  
*CheckMAC command index for key identifier.*
- #define `CHECKMAC_CLIENT_CHALLENGE_IDX ATCA_DATA_IDX`  
*CheckMAC command index for client challenge.*
- #define `CHECKMAC_CLIENT_RESPONSE_IDX` (37)  
*CheckMAC command index for client response.*
- #define `CHECKMAC_DATA_IDX` (69)  
*CheckMAC command index for other data.*
- #define `CHECKMAC_COUNT` (84)  
*CheckMAC command packet size.*
- #define `CHECKMAC_MODE_CHALLENGE` ((uint8\_t)0x00)  
*CheckMAC mode 0: first SHA block from key id.*
- #define `CHECKMAC_MODE_BLOCK2_TEMPKEY` ((uint8\_t)0x01)  
*CheckMAC mode bit 0: second SHA block from TempKey.*
- #define `CHECKMAC_MODE_BLOCK1_TEMPKEY` ((uint8\_t)0x02)  
*CheckMAC mode bit 1: first SHA block from TempKey.*
- #define `CHECKMAC_MODE_SOURCE_FLAG_MATCH` ((uint8\_t)0x04)  
*CheckMAC mode bit 2: match TempKey.SourceFlag.*
- #define `CHECKMAC_MODE_INCLUDE_OTP_64` ((uint8\_t)0x20)  
*CheckMAC mode bit 5: include first 64 OTP bits.*
- #define `CHECKMAC_MODE_MASK` ((uint8\_t)0x27)  
*CheckMAC mode bits 3, 4, 6, and 7 are 0.*
- #define `CHECKMAC_CLIENT_CHALLENGE_SIZE` (32)  
*CheckMAC size of client challenge.*
- #define `CHECKMAC_CLIENT_RESPONSE_SIZE` (32)  
*CheckMAC size of client response.*
- #define `CHECKMAC_OTHER_DATA_SIZE` (13)  
*CheckMAC size of "other data".*
- #define `CHECKMAC_CLIENT_COMMAND_SIZE` (4)  
*CheckMAC size of client command header size inside "other data".*
- #define `CHECKMAC_CMD_MATCH` (0)  
*CheckMAC return value when there is a match.*
- #define `CHECKMAC_CMD_MISMATCH` (1)  
*CheckMAC return value when there is a mismatch.*
- #define `CHECKMAC_RSP_SIZE ATCA_RSP_SIZE_MIN`  
*CheckMAC response packet size.*

#### Definitions for the Counter command

- #define `COUNTER_COUNT ATCA_CMD_SIZE_MIN`
- #define `COUNTER_MODE_IDX ATCA_PARAM1_IDX`  
*Counter command index for mode.*
- #define `COUNTER_KEYID_IDX ATCA_PARAM2_IDX`  
*Counter command index for key id.*
- #define `COUNTER_MODE_MASK` ((uint8\_t)0x01)  
*Counter mode bits 1 to 7 are 0.*
- #define `COUNTER_MAX_VALUE` ((uint32\_t)2097151)  
*Counter maximum value of the counter.*
- #define `COUNTER_MODE_READ` ((uint8\_t)0x00)

- Counter command mode for reading.
- #define COUNTER\_MODE\_INCREMENT ((uint8\_t)0x01)
- Counter command mode for incrementing.
- #define COUNTER\_RSP\_SIZE ATCA\_RSP\_SIZE\_4
- Counter command response packet size.
- #define COUNTER\_SIZE ATCA\_RSP\_SIZE\_MIN
- Counter size in binary.

### Definitions for the DeriveKey Command

- #define DERIVE\_KEY\_RANDOM\_IDX ATCA\_PARAM1\_IDX
- DeriveKey command index for random bit.
- #define DERIVE\_KEY\_TARGETKEY\_IDX ATCA\_PARAM2\_IDX
- DeriveKey command index for target slot.
- #define DERIVE\_KEY\_MAC\_IDX ATCA\_DATA\_IDX
- DeriveKey command index for optional MAC.
- #define DERIVE\_KEY\_COUNT\_SMALL ATCA\_CMD\_SIZE\_MIN
- DeriveKey command packet size without MAC.
- #define DERIVE\_KEY\_MODE ((uint8\_t)0x04)
- DeriveKey command mode set to 4 as in datasheet.
- #define DERIVE\_KEY\_COUNT\_LARGE (39)
- DeriveKey command packet size with MAC.
- #define DERIVE\_KEY\_RANDOM\_FLAG ((uint8\_t)4)
- DeriveKey 1. parameter; has to match TempKey.SourceFlag.
- #define DERIVE\_KEY\_MAC\_SIZE (32)
- DeriveKey MAC size.
- #define DERIVE\_KEY\_RSP\_SIZE ATCA\_RSP\_SIZE\_MIN
- DeriveKey response packet size.

### Definitions for the ECDH Command

- #define ECDH\_PREFIX\_MODE ((uint8\_t)0x00)
- #define ECDH\_COUNT (ATCA\_CMD\_SIZE\_MIN + ATCA\_PUB\_KEY\_SIZE)
- #define ECDH\_MODE\_SOURCE\_MASK ((uint8\_t)0x01)
- #define ECDH\_MODE\_SOURCE\_EEPROM\_SLOT ((uint8\_t)0x00)
- #define ECDH\_MODE\_SOURCE\_TEMPKEY ((uint8\_t)0x01)
- #define ECDH\_MODE\_OUTPUT\_MASK ((uint8\_t)0x02)
- #define ECDH\_MODE\_OUTPUT\_CLEAR ((uint8\_t)0x00)
- #define ECDH\_MODE\_OUTPUT\_ENC ((uint8\_t)0x02)
- #define ECDH\_MODE\_COPY\_MASK ((uint8\_t)0x0C)
- #define ECDH\_MODE\_COPY\_COMPATIBLE ((uint8\_t)0x00)
- #define ECDH\_MODE\_COPY\_EEPROM\_SLOT ((uint8\_t)0x04)
- #define ECDH\_MODE\_COPY\_TEMP\_KEY ((uint8\_t)0x08)
- #define ECDH\_MODE\_COPY\_OUTPUT\_BUFFER ((uint8\_t)0x0C)
- #define ECDH\_KEY\_SIZE ATCA\_BLOCK\_SIZE
- ECDH output data size.
- #define ECDH\_RSP\_SIZE ATCA\_RSP\_SIZE\_64
- ECDH command packet size.

### Definitions for the GenDig Command

- #define GENDIG\_ZONE\_IDX ATCA\_PARAM1\_IDX
- GenDig command index for zone.
- #define GENDIG\_KEYID\_IDX ATCA\_PARAM2\_IDX
- GenDig command index for key id.
- #define GENDIG\_DATA\_IDX ATCA\_DATA\_IDX
- GenDig command index for optional data.
- #define GENDIG\_COUNT ATCA\_CMD\_SIZE\_MIN

- *GenDig command packet size without "other data".*  
• #define [GENDIG\\_ZONE\\_CONFIG](#) ((uint8\_t)0)  
*GenDig zone id config. Use KeyID to specify any of the four 256-bit blocks of the Configuration zone.*
- #define [GENDIG\\_ZONE\\_OTP](#) ((uint8\_t)1)  
*GenDig zone id OTP. Use KeyID to specify either the first or second 256-bit block of the OTP zone.*
- #define [GENDIG\\_ZONE\\_DATA](#) ((uint8\_t)2)  
*GenDig zone id data. Use KeyID to specify a slot in the Data zone or a transport key in the hardware array.*
- #define [GENDIG\\_ZONE\\_SHARED\\_NONCE](#) ((uint8\_t)3)  
*GenDig zone id shared nonce. KeyID specifies the location of the input value in the message generation.*
- #define [GENDIG\\_ZONE\\_COUNTER](#) ((uint8\_t)4)  
*GenDig zone id counter. KeyID specifies the monotonic counter ID to be included in the message generation.*
- #define [GENDIG\\_ZONE\\_KEY\\_CONFIG](#) ((uint8\_t)5)  
*GenDig zone id key config. KeyID specifies the slot for which the configuration information is to be included in the message generation.*
- #define [GENDIG\\_RSP\\_SIZE](#) [ATCA\\_RSP\\_SIZE\\_MIN](#)  
*GenDig command response packet size.*

### Definitions for the GenKey Command

- #define [GENKEY\\_MODE\\_IDX](#) [ATCA\\_PARAM1\\_IDX](#)  
*GenKey command index for mode.*
- #define [GENKEY\\_KEYID\\_IDX](#) [ATCA\\_PARAM2\\_IDX](#)  
*GenKey command index for key id.*
- #define [GENKEY\\_DATA\\_IDX](#) (5)  
*GenKey command index for other data.*
- #define [GENKEY\\_COUNT](#) [ATCA\\_CMD\\_SIZE\\_MIN](#)  
*GenKey command packet size without "other data".*
- #define [GENKEY\\_COUNT\\_DATA](#) (10)  
*GenKey command packet size with "other data".*
- #define [GENKEY\\_OTHER\\_DATA\\_SIZE](#) (3)  
*GenKey size of "other data".*
- #define [GENKEY\\_MODE\\_MASK](#) ((uint8\_t)0x1C)  
*GenKey mode bits 0 to 1 and 5 to 7 are 0.*
- #define [GENKEY\\_MODE\\_PRIVATE](#) ((uint8\_t)0x04)  
*GenKey mode: private key generation.*
- #define [GENKEY\\_MODE\\_PUBLIC](#) ((uint8\_t)0x00)  
*GenKey mode: public key calculation.*
- #define [GENKEY\\_MODE\\_DIGEST](#) ((uint8\_t)0x08)  
*GenKey mode: PubKey digest will be created after the public key is calculated.*
- #define [GENKEY\\_MODE\\_PUBKEY\\_DIGEST](#) ((uint8\_t)0x10)  
*GenKey mode: Calculate PubKey digest on the public key in KeyId.*
- #define [GENKEY\\_PRIVATE\\_TO\\_TEMPKEY](#) ((uint16\_t)0xFFFF)  
*GenKey Create private key and store to tempkey (608 only)*
- #define [GENKEY\\_RSP\\_SIZE\\_SHORT](#) [ATCA\\_RSP\\_SIZE\\_MIN](#)  
*GenKey response packet size in Digest mode.*
- #define [GENKEY\\_RSP\\_SIZE\\_LONG](#) [ATCA\\_RSP\\_SIZE\\_64](#)  
*GenKey response packet size when returning a public key.*

### Definitions for the HMAC Command

- #define [HMAC\\_MODE\\_IDX](#) [ATCA\\_PARAM1\\_IDX](#)  
*HMAC command index for mode.*
- #define [HMAC\\_KEYID\\_IDX](#) [ATCA\\_PARAM2\\_IDX](#)  
*HMAC command index for key id.*
- #define [HMAC\\_COUNT](#) [ATCA\\_CMD\\_SIZE\\_MIN](#)  
*HMAC command packet size.*
- #define [HMAC\\_MODE\\_FLAG\\_TK\\_RAND](#) ((uint8\_t)0x00)

- HMAC mode bit 2: The value of this bit must match the value in TempKey.SourceFlag or the command will return an error.
- #define **HMAC\_MODE\_FLAG\_TK\_NORAND** ((uint8\_t)0x04)  
HMAC mode bit 2: The value of this bit must match the value in TempKey.SourceFlag or the command will return an error.
  - #define **HMAC\_MODE\_FLAG\_OTP88** ((uint8\_t)0x10)  
HMAC mode bit 4: Include the first 88 OTP bits (OTP[0] through OTP[10]) in the message.; otherwise, the corresponding message bits are set to zero. Not applicable for ATECC508A.
  - #define **HMAC\_MODE\_FLAG\_OTP64** ((uint8\_t)0x20)  
HMAC mode bit 5: Include the first 64 OTP bits (OTP[0] through OTP[7]) in the message.; otherwise, the corresponding message bits are set to zero. If Mode[4] is set, the value of this mode bit is ignored. Not applicable for ATECC508A.
  - #define **HMAC\_MODE\_FLAG\_FULLSN** ((uint8\_t)0x40)  
HMAC mode bit 6: If set, include the 48 bits SN[2:3] and SN[4:7] in the message.; otherwise, the corresponding message bits are set to zero.
  - #define **HMAC\_MODE\_MASK** ((uint8\_t)0x74)  
HMAC mode bits 0, 1, 3, and 7 are 0.
  - #define **HMAC\_DIGEST\_SIZE** (32)  
HMAC size of digest response.
  - #define **HMAC\_RSP\_SIZE ATCA\_RSP\_SIZE\_32**  
HMAC command response packet size.

### Definitions for the Info Command

- #define **INFO\_PARAM1\_IDX ATCA\_PARAM1\_IDX**  
Info command index for 1. parameter.
- #define **INFO\_PARAM2\_IDX ATCA\_PARAM2\_IDX**  
Info command index for 2. parameter.
- #define **INFO\_COUNT ATCA\_CMD\_SIZE\_MIN**  
Info command packet size.
- #define **INFO\_MODE\_REVISION** ((uint8\_t)0x00)  
Info mode Revision.
- #define **INFO\_MODE\_KEY\_VALID** ((uint8\_t)0x01)  
Info mode KeyValid.
- #define **INFO\_MODE\_STATE** ((uint8\_t)0x02)  
Info mode State.
- #define **INFO\_MODE\_GPIO** ((uint8\_t)0x03)  
Info mode GPIO.
- #define **INFO\_MODE\_VOL\_KEY\_PERMIT** ((uint8\_t)0x04)  
Info mode GPIO.
- #define **INFO\_MODE\_MAX** ((uint8\_t)0x03)  
Info mode maximum value.
- #define **INFO\_NO\_STATE** ((uint8\_t)0x00)  
Info mode is not the state mode.
- #define **INFO\_OUTPUT\_STATE\_MASK** ((uint8\_t)0x01)  
Info output state mask.
- #define **INFO\_DRIVER\_STATE\_MASK** ((uint8\_t)0x02)  
Info driver state mask.
- #define **INFO\_PARAM2\_SET\_LATCH\_STATE** ((uint16\_t)0x0002)  
Info param2 to set the persistent latch state.
- #define **INFO\_PARAM2\_LATCH\_SET** ((uint16\_t)0x0001)  
Info param2 to set the persistent latch.
- #define **INFO\_PARAM2\_LATCH\_CLEAR** ((uint16\_t)0x0000)  
Info param2 to clear the persistent latch.
- #define **INFO\_SIZE** ((uint8\_t)0x04)  
Info return size.
- #define **INFO\_RSP\_SIZE ATCA\_RSP\_SIZE\_VAL**  
Info command response packet size.

**Definitions for the KDF Command**

- #define `KDF_MODE_IDX ATCA_PARAM1_IDX`  
*KDF command index for mode.*
- #define `KDF_KEYID_IDX ATCA_PARAM2_IDX`  
*KDF command index for key id.*
- #define `KDF_DETAILS_IDX ATCA_DATA_IDX`  
*KDF command index for details.*
- #define `KDF_DETAILS_SIZE 4`  
*KDF details (param3) size.*
- #define `KDF_MESSAGE_IDX (ATCA_DATA_IDX + KDF_DETAILS_SIZE)`
- #define `KDF_MODE_SOURCE_MASK ((uint8_t)0x03)`  
*KDF mode source key mask.*
- #define `KDF_MODE_SOURCE_TEMPKEY ((uint8_t)0x00)`  
*KDF mode source key in TempKey.*
- #define `KDF_MODE_SOURCE_TEMPKEY_UP ((uint8_t)0x01)`  
*KDF mode source key in upper TempKey.*
- #define `KDF_MODE_SOURCE_SLOT ((uint8_t)0x02)`  
*KDF mode source key in a slot.*
- #define `KDF_MODE_SOURCE_ALTKEYBUF ((uint8_t)0x03)`  
*KDF mode source key in alternate key buffer.*
- #define `KDF_MODE_TARGET_MASK ((uint8_t)0x1C)`  
*KDF mode target key mask.*
- #define `KDF_MODE_TARGET_TEMPKEY ((uint8_t)0x00)`  
*KDF mode target key in TempKey.*
- #define `KDF_MODE_TARGET_TEMPKEY_UP ((uint8_t)0x04)`  
*KDF mode target key in upper TempKey.*
- #define `KDF_MODE_TARGET_SLOT ((uint8_t)0x08)`  
*KDF mode target key in slot.*
- #define `KDF_MODE_TARGET_ALTKEYBUF ((uint8_t)0x0C)`  
*KDF mode target key in alternate key buffer.*
- #define `KDF_MODE_TARGET_OUTPUT ((uint8_t)0x10)`  
*KDF mode target key in output buffer.*
- #define `KDF_MODE_TARGET_OUTPUT_ENC ((uint8_t)0x14)`  
*KDF mode target key encrypted in output buffer.*
- #define `KDF_MODE_ALG_MASK ((uint8_t)0x60)`  
*KDF mode algorithm mask.*
- #define `KDF_MODE_ALG_PRF ((uint8_t)0x00)`  
*KDF mode PRF algorithm.*
- #define `KDF_MODE_ALG_AES ((uint8_t)0x20)`  
*KDF mode AES algorithm.*
- #define `KDF_MODE_ALG_HKDF ((uint8_t)0x40)`  
*KDF mode HKDF algorithm.*
- #define `KDF_DETAILS_PRF_KEY_LEN_MASK ((uint32_t)0x00000003)`  
*KDF details for PRF, source key length mask.*
- #define `KDF_DETAILS_PRF_KEY_LEN_16 ((uint32_t)0x00000000)`  
*KDF details for PRF, source key length is 16 bytes.*
- #define `KDF_DETAILS_PRF_KEY_LEN_32 ((uint32_t)0x00000001)`  
*KDF details for PRF, source key length is 32 bytes.*
- #define `KDF_DETAILS_PRF_KEY_LEN_48 ((uint32_t)0x00000002)`  
*KDF details for PRF, source key length is 48 bytes.*
- #define `KDF_DETAILS_PRF_KEY_LEN_64 ((uint32_t)0x00000003)`  
*KDF details for PRF, source key length is 64 bytes.*
- #define `KDF_DETAILS_PRF_TARGET_LEN_MASK ((uint32_t)0x00000100)`  
*KDF details for PRF, target length mask.*
- #define `KDF_DETAILS_PRF_TARGET_LEN_32 ((uint32_t)0x00000000)`  
*KDF details for PRF, target length is 32 bytes.*
- #define `KDF_DETAILS_PRF_TARGET_LEN_64 ((uint32_t)0x00000100)`  
*KDF details for PRF, target length is 64 bytes.*

- #define `KDF_DETAILS_PRF_AEAD_MASK` ((uint32\_t)0x00000600)  
*KDF details for PRF, AEAD processing mask.*
- #define `KDF_DETAILS_PRF_AEAD_MODE0` ((uint32\_t)0x00000000)  
*KDF details for PRF, AEAD no processing.*
- #define `KDF_DETAILS_PRF_AEAD_MODE1` ((uint32\_t)0x00000200)  
*KDF details for PRF, AEAD First 32 go to target, second 32 go to output buffer.*
- #define `KDF_DETAILS_AES_KEY_LOC_MASK` ((uint32\_t)0x00000003)  
*KDF details for AES, key location mask.*
- #define `KDF_DETAILS_HKDF_MSG_LOC_MASK` ((uint32\_t)0x00000003)  
*KDF details for HKDF, message location mask.*
- #define `KDF_DETAILS_HKDF_MSG_LOC_SLOT` ((uint32\_t)0x00000000)  
*KDF details for HKDF, message location in slot.*
- #define `KDF_DETAILS_HKDF_MSG_LOC_TEMPKEY` ((uint32\_t)0x00000001)  
*KDF details for HKDF, message location in TempKey.*
- #define `KDF_DETAILS_HKDF_MSG_LOC_INPUT` ((uint32\_t)0x00000002)  
*KDF details for HKDF, message location in input parameter.*
- #define `KDF_DETAILS_HKDF_MSG_LOC_IV` ((uint32\_t)0x00000003)  
*KDF details for HKDF, message location is a special IV function.*
- #define `KDF_DETAILS_HKDF_ZERO_KEY` ((uint32\_t)0x00000004)  
*KDF details for HKDF, key is 32 bytes of zero.*

### Definitions for the Lock Command

- #define `LOCK_ZONE_IDX ATCA_PARAM1_IDX`  
*Lock command index for zone.*
- #define `LOCK_SUMMARY_IDX ATCA_PARAM2_IDX`  
*Lock command index for summary.*
- #define `LOCK_COUNT ATCA_CMD_SIZE_MIN`  
*Lock command packet size.*
- #define `LOCK_ZONE_CONFIG` ((uint8\_t)0x00)  
*Lock zone is Config.*
- #define `LOCK_ZONE_DATA` ((uint8\_t)0x01)  
*Lock zone is OTP or Data.*
- #define `LOCK_ZONE_DATA_SLOT` ((uint8\_t)0x02)  
*Lock slot of Data.*
- #define `LOCK_ZONE_NO_CRC` ((uint8\_t)0x80)  
*Lock command: Ignore summary.*
- #define `LOCK_ZONE_MASK` (0xBF)  
*Lock parameter 1 bits 6 are 0.*
- #define `ATCA_UNLOCKED` (0x55)  
*Value indicating an unlocked zone.*
- #define `ATCA_LOCKED` (0x00)  
*Value indicating a locked zone.*
- #define `LOCK_RSP_SIZE ATCA_RSP_SIZE_MIN`  
*Lock command response packet size.*

### Definitions for the MAC Command

- #define `MAC_MODE_IDX ATCA_PARAM1_IDX`  
*MAC command index for mode.*
- #define `MAC_KEYID_IDX ATCA_PARAM2_IDX`  
*MAC command index for key id.*
- #define `MAC_CHALLENGE_IDX ATCA_DATA_IDX`  
*MAC command index for optional challenge.*
- #define `MAC_COUNT_SHORT ATCA_CMD_SIZE_MIN`  
*MAC command packet size without challenge.*
- #define `MAC_COUNT_LONG` (39)  
*MAC command packet size with challenge.*



- #define `MAC_MODE_CHALLENGE` ((uint8\_t)0x00)  
*MAC mode 0: first SHA block from data slot.*
- #define `MAC_MODE_BLOCK2_TEMPKEY` ((uint8\_t)0x01)  
*MAC mode bit 0: second SHA block from TempKey.*
- #define `MAC_MODE_BLOCK1_TEMPKEY` ((uint8\_t)0x02)  
*MAC mode bit 1: first SHA block from TempKey.*
- #define `MAC_MODE_SOURCE_FLAG_MATCH` ((uint8\_t)0x04)  
*MAC mode bit 2: match TempKey.SourceFlag.*
- #define `MAC_MODE_PTNonce_TEMPKEY` ((uint8\_t)0x06)  
*MAC mode bit 0: second SHA block from TempKey.*
- #define `MAC_MODE_PASSTHROUGH` ((uint8\_t)0x07)  
*MAC mode bit 0-2: pass-through mode.*
- #define `MAC_MODE_INCLUDE_OTP_88` ((uint8\_t)0x10)  
*MAC mode bit 4: include first 88 OTP bits.*
- #define `MAC_MODE_INCLUDE_OTP_64` ((uint8\_t)0x20)  
*MAC mode bit 5: include first 64 OTP bits.*
- #define `MAC_MODE_INCLUDE_SN` ((uint8\_t)0x40)  
*MAC mode bit 6: include serial number.*
- #define `MAC_CHALLENGE_SIZE` (32)  
*MAC size of challenge.*
- #define `MAC_SIZE` (32)  
*MAC size of response.*
- #define `MAC_MODE_MASK` ((uint8\_t)0x77)  
*MAC mode bits 3 and 7 are 0.*
- #define `MAC_RSP_SIZE` `ATCA_RSP_SIZE_32`  
*MAC command response packet size.*

#### Definitions for the Nonce Command

- #define `NONCE_MODE_IDX` `ATCA_PARAM1_IDX`  
*Nonce command index for mode.*
- #define `NONCE_PARAM2_IDX` `ATCA_PARAM2_IDX`  
*Nonce command index for 2. parameter.*
- #define `NONCE_INPUT_IDX` `ATCA_DATA_IDX`  
*Nonce command index for input data.*
- #define `NONCE_COUNT_SHORT` (`ATCA_CMD_SIZE_MIN` + 20)  
*Nonce command packet size for 20 bytes of NumIn.*
- #define `NONCE_COUNT_LONG` (`ATCA_CMD_SIZE_MIN` + 32)  
*Nonce command packet size for 32 bytes of NumIn.*
- #define `NONCE_COUNT_LONG_64` (`ATCA_CMD_SIZE_MIN` + 64)  
*Nonce command packet size for 64 bytes of NumIn.*
- #define `NONCE_MODE_MASK` ((uint8\_t)0x03)  
*Nonce mode bits 2 to 7 are 0.*
- #define `NONCE_MODE_SEED_UPDATE` ((uint8\_t)0x00)  
*Nonce mode: update seed.*
- #define `NONCE_MODE_NO_SEED_UPDATE` ((uint8\_t)0x01)  
*Nonce mode: do not update seed.*
- #define `NONCE_MODE_INVALID` ((uint8\_t)0x02)  
*Nonce mode 2 is invalid.*
- #define `NONCE_MODE_PASSTHROUGH` ((uint8\_t)0x03)  
*Nonce mode: pass-through.*
- #define `NONCE_MODE_INPUT_LEN_MASK` ((uint8\_t)0x20)  
*Nonce mode: input size mask.*
- #define `NONCE_MODE_INPUT_LEN_32` ((uint8\_t)0x00)  
*Nonce mode: input size is 32 bytes.*
- #define `NONCE_MODE_INPUT_LEN_64` ((uint8\_t)0x20)  
*Nonce mode: input size is 64 bytes.*
- #define `NONCE_MODE_TARGET_MASK` ((uint8\_t)0xC0)



- *Nonce mode: target mask.*
- #define `NONCE_MODE_TARGET_TEMPKEY` ((uint8\_t)0x00)
- *Nonce mode: target is TempKey.*
- #define `NONCE_MODE_TARGET_MSGDIGBUF` ((uint8\_t)0x40)
- *Nonce mode: target is Message Digest Buffer.*
- #define `NONCE_MODE_TARGET_ALTKEYBUF` ((uint8\_t)0x80)
- *Nonce mode: target is Alternate Key Buffer.*
- #define `NONCE_ZERO_CALC_MASK` ((uint16\_t)0x8000)
- *Nonce zero (param2): calculation mode mask.*
- #define `NONCE_ZERO_CALC_RANDOM` ((uint16\_t)0x0000)
- *Nonce zero (param2): calculation mode random, use RNG in calculation and return RNG output.*
- #define `NONCE_ZERO_CALC_TEMPKEY` ((uint16\_t)0x8000)
- *Nonce zero (param2): calculation mode TempKey, use TempKey in calculation and return new TempKey value.*
- #define `NONCE_NUMIN_SIZE` (20)
- *Nonce NumIn size for random modes.*
- #define `NONCE_NUMIN_SIZE_PASSTHROUGH` (32)
- *Nonce NumIn size for 32-byte pass-through mode.*
- #define `NONCE_RSP_SIZE_SHORT ATCA_RSP_SIZE_MIN`
- *Nonce command response packet size with no output.*
- #define `NONCE_RSP_SIZE_LONG ATCA_RSP_SIZE_32`
- *Nonce command response packet size with output.*

#### Definitions for the Pause Command

- #define `PAUSE_SELECT_IDX ATCA_PARAM1_IDX`
- *Pause command index for Selector.*
- #define `PAUSE_PARAM2_IDX ATCA_PARAM2_IDX`
- *Pause command index for 2. parameter.*
- #define `PAUSE_COUNT ATCA_CMD_SIZE_MIN`
- *Pause command packet size.*
- #define `PAUSE_RSP_SIZE ATCA_RSP_SIZE_MIN`
- *Pause command response packet size.*

#### Definitions for the PrivWrite Command

- #define `PRIVWRITE_ZONE_IDX ATCA_PARAM1_IDX`
- *PrivWrite command index for zone.*
- #define `PRIVWRITE_KEYID_IDX ATCA_PARAM2_IDX`
- *PrivWrite command index for KeyID.*
- #define `PRIVWRITE_VALUE_IDX` ( 5)
- *PrivWrite command index for value.*
- #define `PRIVWRITE_MAC_IDX` (41)
- *PrivWrite command index for MAC.*
- #define `PRIVWRITE_COUNT` (75)
- *PrivWrite command packet size.*
- #define `PRIVWRITE_ZONE_MASK` ((uint8\_t)0x40)
- *PrivWrite zone bits 0 to 5 and 7 are 0.*
- #define `PRIVWRITE_MODE_ENCRYPT` ((uint8\_t)0x40)
- *PrivWrite mode: encrypted.*
- #define `PRIVWRITE_RSP_SIZE ATCA_RSP_SIZE_MIN`
- *PrivWrite command response packet size.*

#### Definitions for the Random Command

- #define `RANDOM_MODE_IDX ATCA_PARAM1_IDX`
- *Random command index for mode.*
- #define `RANDOM_PARAM2_IDX ATCA_PARAM2_IDX`

- *Random command index for 2. parameter.*  
• #define [RANDOM\\_COUNT ATCA\\_CMD\\_SIZE\\_MIN](#)
- *Random command packet size.*  
• #define [RANDOM\\_SEED\\_UPDATE](#) ((uint8\_t)0x00)
- *Random mode for automatic seed update.*  
• #define [RANDOM\\_NO\\_SEED\\_UPDATE](#) ((uint8\_t)0x01)
- *Random mode for no seed update.*  
• #define [RANDOM\\_NUM\\_SIZE](#) ((uint8\_t)32)
- *Number of bytes in the data packet of a random command.*  
• #define [RANDOM\\_RSP\\_SIZE ATCA\\_RSP\\_SIZE\\_32](#)
- *Random command response packet size.*

#### Definitions for the Read Command

- #define [READ\\_ZONE\\_IDX ATCA\\_PARAM1\\_IDX](#)  
*Read command index for zone.*
- #define [READ\\_ADDR\\_IDX ATCA\\_PARAM2\\_IDX](#)  
*Read command index for address.*
- #define [READ\\_COUNT ATCA\\_CMD\\_SIZE\\_MIN](#)  
*Read command packet size.*
- #define [READ\\_ZONE\\_MASK](#) ((uint8\_t)0x83)  
*Read zone bits 2 to 6 are 0.*
- #define [READ\\_4\\_RSP\\_SIZE ATCA\\_RSP\\_SIZE\\_VAL](#)  
*Read command response packet size when reading 4 bytes.*
- #define [READ\\_32\\_RSP\\_SIZE ATCA\\_RSP\\_SIZE\\_32](#)  
*Read command response packet size when reading 32 bytes.*

#### Definitions for the SecureBoot Command

- #define [SECUREBOOT\\_MODE\\_IDX ATCA\\_PARAM1\\_IDX](#)  
*SecureBoot command index for mode.*
- #define [SECUREBOOT\\_DIGEST\\_SIZE](#) (32)  
*SecureBoot digest input size.*
- #define [SECUREBOOT\\_SIGNATURE\\_SIZE](#) (64)  
*SecureBoot signature input size.*
- #define [SECUREBOOT\\_COUNT\\_DIG](#) (ATCA\_CMD\_SIZE\_MIN + SECUREBOOT\_DIGEST\_SIZE)  
*SecureBoot command packet size for just a digest.*
- #define [SECUREBOOT\\_COUNT\\_DIG\\_SIG](#) (ATCA\_CMD\_SIZE\_MIN + SECUREBOOT\_DIGEST\_SIZE + SECUREBOOT\_SIGNATURE\_SIZE)  
*SecureBoot command packet size for a digest and signature.*
- #define [SECUREBOOT\\_MAC\\_SIZE](#) (32)  
*SecureBoot MAC output size.*
- #define [SECUREBOOT\\_RSP\\_SIZE\\_NO\\_MAC ATCA\\_RSP\\_SIZE\\_MIN](#)  
*SecureBoot response packet size for no MAC.*
- #define [SECUREBOOT\\_RSP\\_SIZE\\_MAC](#) (ATCA\_PACKET\_OVERHEAD + SECUREBOOT\_MAC\_SIZE)  
*SecureBoot response packet size with MAC.*
- #define [SECUREBOOT\\_MODE\\_MASK](#) ((uint8\_t)0x07)  
*SecureBoot mode mask.*
- #define [SECUREBOOT\\_MODE\\_FULL](#) ((uint8\_t)0x05)  
*SecureBoot mode Full.*
- #define [SECUREBOOT\\_MODE\\_FULL\\_STORE](#) ((uint8\_t)0x06)  
*SecureBoot mode FullStore.*
- #define [SECUREBOOT\\_MODE\\_FULL\\_COPY](#) ((uint8\_t)0x07)  
*SecureBoot mode FullCopy.*
- #define [SECUREBOOT\\_MODE\\_PROHIBIT\\_FLAG](#) ((uint8\_t)0x40)  
*SecureBoot mode flag to prohibit SecureBoot until next power cycle.*
- #define [SECUREBOOT\\_MODE\\_ENC\\_MAC\\_FLAG](#) ((uint8\_t)0x80)  
*SecureBoot mode flag for encrypted digest and returning validating MAC.*

- #define `SECUREBOOTCONFIG_OFFSET` (70)  
*SecureBootConfig byte offset into the configuration zone.*
- #define `SECUREBOOTCONFIG_MODE_MASK` ((uint16\_t)0x0003)  
*Mask for SecureBootMode field in SecureBootConfig value.*
- #define `SECUREBOOTCONFIG_MODE_DISABLED` ((uint16\_t)0x0000)  
*Disabled SecureBootMode in SecureBootConfig value.*
- #define `SECUREBOOTCONFIG_MODE_FULL_BOTH` ((uint16\_t)0x0001)  
*Both digest and signature always required SecureBootMode in SecureBootConfig value.*
- #define `SECUREBOOTCONFIG_MODE_FULL_SIG` ((uint16\_t)0x0002)  
*Signature stored SecureBootMode in SecureBootConfig value.*
- #define `SECUREBOOTCONFIG_MODE_FULL_DIG` ((uint16\_t)0x0003)  
*Digest stored SecureBootMode in SecureBootConfig value.*

### Definitions for the SelfTest Command

- #define `SELFTEST_MODE_IDX ATCA_PARAM1_IDX`  
*SelfTest command index for mode.*
- #define `SELFTEST_COUNT ATCA_CMD_SIZE_MIN`  
*SelfTest command packet size.*
- #define `SELFTEST_MODE_RNG` ((uint8\_t)0x01)  
*SelfTest mode RNG DRBG function.*
- #define `SELFTEST_MODE_ECDSA_SIGN_VERIFY` ((uint8\_t)0x02)  
*SelfTest mode ECDSA verify function.*
- #define `SELFTEST_MODE_ECDH` ((uint8\_t)0x08)  
*SelfTest mode ECDH function.*
- #define `SELFTEST_MODE_AES` ((uint8\_t)0x10)  
*SelfTest mode AES encrypt function.*
- #define `SELFTEST_MODE_SHA` ((uint8\_t)0x20)  
*SelfTest mode SHA function.*
- #define `SELFTEST_MODE_ALL` ((uint8\_t)0x3B)  
*SelfTest mode all algorithms.*
- #define `SELFTEST_RSP_SIZE ATCA_RSP_SIZE_MIN`  
*SelfTest command response packet size.*

### Definitions for the SHA Command

- #define `SHA_COUNT_SHORT ATCA_CMD_SIZE_MIN`
- #define `SHA_COUNT_LONG ATCA_CMD_SIZE_MIN`  
*Just a starting size.*
- #define `ATCA_SHA_DIGEST_SIZE` (32)
- #define `SHA_DATA_MAX` (64)
- #define `SHA_MODE_MASK` ((uint8\_t)0x07)  
*Mask the bit 0-2.*
- #define `SHA_MODE_SHA256_START` ((uint8\_t)0x00)  
*Initialization, does not accept a message.*
- #define `SHA_MODE_SHA256_UPDATE` ((uint8\_t)0x01)  
*Add 64 bytes in the message to the SHA context.*
- #define `SHA_MODE_SHA256_END` ((uint8\_t)0x02)  
*Complete the calculation and return the digest.*
- #define `SHA_MODE_SHA256_PUBLIC` ((uint8\_t)0x03)  
*Add 64 byte ECC public key in the slot to the SHA context.*
- #define `SHA_MODE_HMAC_START` ((uint8\_t)0x04)  
*Initialization, HMAC calculation.*
- #define `SHA_MODE_HMAC_UPDATE` ((uint8\_t)0x05)  
*Add 64 bytes in the message to the SHA context.*
- #define `SHA_MODE_HMAC_END` ((uint8\_t)0x06)  
*Complete the HMAC computation and return digest.*
- #define `SHA_MODE_608_HMAC_END` ((uint8\_t)0x02)

- Complete the HMAC computation and return digest... Different command on 608.
- #define [SHA\\_MODE\\_READ\\_CONTEXT](#) ((uint8\_t)0x06)  
*Read current SHA-256 context out of the device.*
- #define [SHA\\_MODE\\_WRITE\\_CONTEXT](#) ((uint8\_t)0x07)  
*Restore a SHA-256 context into the device.*
- #define [SHA\\_MODE\\_TARGET\\_MASK](#) ((uint8\_t)0xC0)  
*Resulting digest target location mask.*
- #define [SHA\\_RSP\\_SIZE](#) [ATCA\\_RSP\\_SIZE\\_32](#)  
*SHA command response packet size.*
- #define [SHA\\_RSP\\_SIZE\\_SHORT](#) [ATCA\\_RSP\\_SIZE\\_MIN](#)  
*SHA command response packet size only status code.*
- #define [SHA\\_RSP\\_SIZE\\_LONG](#) [ATCA\\_RSP\\_SIZE\\_32](#)  
*SHA command response packet size.*

### Definitions for the Sign Command

- #define [SIGN\\_MODE\\_IDX](#) [ATCA\\_PARAM1\\_IDX](#)  
*Sign command index for mode.*
- #define [SIGN\\_KEYID\\_IDX](#) [ATCA\\_PARAM2\\_IDX](#)  
*Sign command index for key id.*
- #define [SIGN\\_COUNT](#) [ATCA\\_CMD\\_SIZE\\_MIN](#)  
*Sign command packet size.*
- #define [SIGN\\_MODE\\_MASK](#) ((uint8\_t)0xE1)  
*Sign mode bits 1 to 4 are 0.*
- #define [SIGN\\_MODE\\_INTERNAL](#) ((uint8\_t)0x00)  
*Sign mode 0: internal.*
- #define [SIGN\\_MODE\\_INVALIDATE](#) ((uint8\_t)0x01)  
*Sign mode bit 1: Signature will be used for Verify(Invalidate)*
- #define [SIGN\\_MODE\\_INCLUDE\\_SN](#) ((uint8\_t)0x40)  
*Sign mode bit 6: include serial number.*
- #define [SIGN\\_MODE\\_EXTERNAL](#) ((uint8\_t)0x80)  
*Sign mode bit 7: external.*
- #define [SIGN\\_MODE\\_SOURCE\\_MASK](#) ((uint8\_t)0x20)  
*Sign mode message source mask.*
- #define [SIGN\\_MODE\\_SOURCE\\_TEMPKEY](#) ((uint8\_t)0x00)  
*Sign mode message source is TempKey.*
- #define [SIGN\\_MODE\\_SOURCE\\_MSGDIGBUF](#) ((uint8\_t)0x20)  
*Sign mode message source is the Message Digest Buffer.*
- #define [SIGN\\_RSP\\_SIZE](#) [ATCA\\_RSP\\_SIZE\\_MAX](#)  
*Sign command response packet size.*

### Definitions for the UpdateExtra Command

- #define [UPDATE\\_MODE\\_IDX](#) [ATCA\\_PARAM1\\_IDX](#)  
*UpdateExtra command index for mode.*
- #define [UPDATE\\_VALUE\\_IDX](#) [ATCA\\_PARAM2\\_IDX](#)  
*UpdateExtra command index for new value.*
- #define [UPDATE\\_COUNT](#) [ATCA\\_CMD\\_SIZE\\_MIN](#)  
*UpdateExtra command packet size.*
- #define [UPDATE\\_MODE\\_USER\\_EXTRA](#) ((uint8\_t)0x00)  
*UpdateExtra mode update UserExtra (config byte 84)*
- #define [UPDATE\\_MODE\\_SELECTOR](#) ((uint8\_t)0x01)  
*UpdateExtra mode update Selector (config byte 85)*
- #define [UPDATE\\_MODE\\_USER\\_EXTRA\\_ADD](#) [UPDATE\\_MODE\\_SELECTOR](#)  
*UpdateExtra mode update UserExtraAdd (config byte 85)*
- #define [UPDATE\\_MODE\\_DEC\\_COUNTER](#) ((uint8\_t)0x02)  
*UpdateExtra mode: decrement counter.*
- #define [UPDATE\\_RSP\\_SIZE](#) [ATCA\\_RSP\\_SIZE\\_MIN](#)

*UpdateExtra command response packet size.*

## Definitions for the Verify Command

- #define [VERIFY\\_MODE\\_IDX](#) [ATCA\\_PARAM1\\_IDX](#)  
*Verify command index for mode.*
- #define [VERIFY\\_KEYID\\_IDX](#) [ATCA\\_PARAM2\\_IDX](#)  
*Verify command index for key id.*
- #define [VERIFY\\_DATA\\_IDX](#) ( 5)  
*Verify command index for data.*
- #define [VERIFY\\_256\\_STORED\\_COUNT](#) ( 71)  
*Verify command packet size for 256-bit key in stored mode.*
- #define [VERIFY\\_283\\_STORED\\_COUNT](#) ( 79)  
*Verify command packet size for 283-bit key in stored mode.*
- #define [VERIFY\\_256\\_VALIDATE\\_COUNT](#) ( 90)  
*Verify command packet size for 256-bit key in validate mode.*
- #define [VERIFY\\_283\\_VALIDATE\\_COUNT](#) ( 98)  
*Verify command packet size for 283-bit key in validate mode.*
- #define [VERIFY\\_256\\_EXTERNAL\\_COUNT](#) (135)  
*Verify command packet size for 256-bit key in external mode.*
- #define [VERIFY\\_283\\_EXTERNAL\\_COUNT](#) (151)  
*Verify command packet size for 283-bit key in external mode.*
- #define [VERIFY\\_256\\_KEY\\_SIZE](#) ( 64)  
*Verify key size for 256-bit key.*
- #define [VERIFY\\_283\\_KEY\\_SIZE](#) ( 72)  
*Verify key size for 283-bit key.*
- #define [VERIFY\\_256\\_SIGNATURE\\_SIZE](#) ( 64)  
*Verify signature size for 256-bit key.*
- #define [VERIFY\\_283\\_SIGNATURE\\_SIZE](#) ( 72)  
*Verify signature size for 283-bit key.*
- #define [VERIFY\\_OTHER\\_DATA\\_SIZE](#) ( 19)  
*Verify size of "other data".*
- #define [VERIFY\\_MODE\\_MASK](#) ((uint8\_t)0x07)  
*Verify mode bits 3 to 7 are 0.*
- #define [VERIFY\\_MODE\\_STORED](#) ((uint8\_t)0x00)  
*Verify mode: stored.*
- #define [VERIFY\\_MODE\\_VALIDATE\\_EXTERNAL](#) ((uint8\_t)0x01)  
*Verify mode: validate external.*
- #define [VERIFY\\_MODE\\_EXTERNAL](#) ((uint8\_t)0x02)  
*Verify mode: external.*
- #define [VERIFY\\_MODE\\_VALIDATE](#) ((uint8\_t)0x03)  
*Verify mode: validate.*
- #define [VERIFY\\_MODE\\_INVALIDATE](#) ((uint8\_t)0x07)  
*Verify mode: invalidate.*
- #define [VERIFY\\_MODE\\_SOURCE\\_MASK](#) ((uint8\_t)0x20)  
*Verify mode message source mask.*
- #define [VERIFY\\_MODE\\_SOURCE\\_TEMPKEY](#) ((uint8\_t)0x00)  
*Verify mode message source is TempKey.*
- #define [VERIFY\\_MODE\\_SOURCE\\_MSGDIGBUF](#) ((uint8\_t)0x20)  
*Verify mode message source is the Message Digest Buffer.*
- #define [VERIFY\\_MODE\\_MAC\\_FLAG](#) ((uint8\_t)0x80)  
*Verify mode: MAC.*
- #define [VERIFY\\_KEY\\_B283](#) ((uint16\_t)0x0000)  
*Verify key type: B283.*
- #define [VERIFY\\_KEY\\_K283](#) ((uint16\_t)0x0001)  
*Verify key type: K283.*
- #define [VERIFY\\_KEY\\_P256](#) ((uint16\_t)0x0004)  
*Verify key type: P256.*

- `#define VERIFY_RSP_SIZE ATCA_RSP_SIZE_MIN`  
*Verify command response packet size.*
- `#define VERIFY_RSP_SIZE_MAC ATCA_RSP_SIZE_32`  
*Verify command response packet size with validating MAC.*

### Definitions for the Write Command

- `#define WRITE_ZONE_IDX ATCA_PARAM1_IDX`  
*Write command index for zone.*
- `#define WRITE_ADDR_IDX ATCA_PARAM2_IDX`  
*Write command index for address.*
- `#define WRITE_VALUE_IDX ATCA_DATA_IDX`  
*Write command index for data.*
- `#define WRITE_MAC_VS_IDX ( 9)`  
*Write command index for MAC following short data.*
- `#define WRITE_MAC_VL_IDX (37)`  
*Write command index for MAC following long data.*
- `#define WRITE_MAC_SIZE (32)`  
*Write MAC size.*
- `#define WRITE_ZONE_MASK ((uint8_t)0xC3)`  
*Write zone bits 2 to 5 are 0.*
- `#define WRITE_ZONE_WITH_MAC ((uint8_t)0x40)`  
*Write zone bit 6: write encrypted with MAC.*
- `#define WRITE_ZONE_OTP ((uint8_t)1)`  
*Write zone id OTP.*
- `#define WRITE_ZONE_DATA ((uint8_t)2)`  
*Write zone id data.*
- `#define WRITE_RSP_SIZE ATCA_RSP_SIZE_MIN`  
*Write command response packet size.*

### Functions

- `ATCA_STATUS atCheckMAC (ATCACommand ca_cmd, ATCAPacket *packet)`  
*ATCACommand CheckMAC method.*
- `ATCA_STATUS atCounter (ATCACommand ca_cmd, ATCAPacket *packet)`  
*ATCACommand Counter method.*
- `ATCA_STATUS atDeriveKey (ATCACommand ca_cmd, ATCAPacket *packet, bool has_mac)`  
*ATCACommand DeriveKey method.*
- `ATCA_STATUS atECDH (ATCACommand ca_cmd, ATCAPacket *packet)`  
*ATCACommand ECDH method.*
- `ATCA_STATUS atGenDig (ATCACommand ca_cmd, ATCAPacket *packet, bool is_no_mac_key)`  
*ATCACommand Generate Digest method.*
- `ATCA_STATUS atGenKey (ATCACommand ca_cmd, ATCAPacket *packet)`  
*ATCACommand Generate Key method.*
- `ATCA_STATUS atHMAC (ATCACommand ca_cmd, ATCAPacket *packet)`  
*ATCACommand HMAC method.*
- `ATCA_STATUS atInfo (ATCACommand ca_cmd, ATCAPacket *packet)`  
*ATCACommand Info method.*
- `ATCA_STATUS atLock (ATCACommand ca_cmd, ATCAPacket *packet)`  
*ATCACommand Lock method.*
- `ATCA_STATUS atMAC (ATCACommand ca_cmd, ATCAPacket *packet)`  
*ATCACommand MAC method.*
- `ATCA_STATUS atNonce (ATCACommand ca_cmd, ATCAPacket *packet)`

- ATCACommand Nonce method.*
- [ATCA\\_STATUS atPause](#) ([ATCACommand ca\\_cmd](#), [ATCAPacket \\*packet](#))  
*ATCACommand Pause method.*
- [ATCA\\_STATUS atPrivWrite](#) ([ATCACommand ca\\_cmd](#), [ATCAPacket \\*packet](#))  
*ATCACommand PrivWrite method.*
- [ATCA\\_STATUS atRandom](#) ([ATCACommand ca\\_cmd](#), [ATCAPacket \\*packet](#))  
*ATCACommand Random method.*
- [ATCA\\_STATUS atRead](#) ([ATCACommand ca\\_cmd](#), [ATCAPacket \\*packet](#))  
*ATCACommand Read method.*
- [ATCA\\_STATUS atSecureBoot](#) ([ATCACommand ca\\_cmd](#), [ATCAPacket \\*packet](#))  
*ATCACommand SecureBoot method.*
- [ATCA\\_STATUS atSHA](#) ([ATCACommand ca\\_cmd](#), [ATCAPacket \\*packet](#), [uint16\\_t write\\_context\\_size](#))  
*ATCACommand SHA method.*
- [ATCA\\_STATUS atSign](#) ([ATCACommand ca\\_cmd](#), [ATCAPacket \\*packet](#))  
*ATCACommand Sign method.*
- [ATCA\\_STATUS atUpdateExtra](#) ([ATCACommand ca\\_cmd](#), [ATCAPacket \\*packet](#))  
*ATCACommand UpdateExtra method.*
- [ATCA\\_STATUS atVerify](#) ([ATCACommand ca\\_cmd](#), [ATCAPacket \\*packet](#))  
*ATCACommand ECDSA Verify method.*
- [ATCA\\_STATUS atWrite](#) ([ATCACommand ca\\_cmd](#), [ATCAPacket \\*packet](#), [bool has\\_mac](#))  
*ATCACommand Write method.*
- [ATCA\\_STATUS atAES](#) ([ATCACommand ca\\_cmd](#), [ATCAPacket \\*packet](#))  
*ATCACommand AES method.*
- [ATCA\\_STATUS atSelfTest](#) ([ATCACommand ca\\_cmd](#), [ATCAPacket \\*packet](#))  
*ATCACommand AES method.*
- [ATCA\\_STATUS atKDF](#) ([ATCACommand ca\\_cmd](#), [ATCAPacket \\*packet](#))  
*ATCACommand KDF method.*
- [bool atIsSHAFamily](#) ([ATCADeviceType device\\_type](#))  
*determines if a given device type is a SHA device or a superset of a SHA device*
- [bool atIsECCFamily](#) ([ATCADeviceType device\\_type](#))  
*determines if a given device type is an ECC device or a superset of a ECC device*
- [ATCA\\_STATUS isATCAError](#) ([uint8\\_t \\*data](#))  
*checks for basic error frame in data*
- [void atCRC](#) ([size\\_t length](#), [const uint8\\_t \\*data](#), [uint8\\_t \\*crc\\_le](#))  
*Calculates CRC over the given raw data and returns the CRC in little-endian byte order.*
- [void atCalcCrc](#) ([ATCAPacket \\*pkt](#))  
*This function calculates CRC and adds it to the correct offset in the packet data.*
- [ATCA\\_STATUS atCheckCrc](#) ([const uint8\\_t \\*response](#))  
*This function checks the consistency of a response.*

### 20.71.1 Detailed Description

Microchip Crypto Auth device command object - this is a command builder only, it does not send the command. The result of a command method is a fully formed packet, ready to send to the ATCAIFace object to dispatch.

This command object supports the ATSHA and ATECC device family. The command list is a superset of all device commands for this family. The command object differentiates the packet contents based on specific device type within the family.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.71.2 Macro Definition Documentation

#### 20.71.2.1 AES\_COUNT

```
#define AES_COUNT (23)
```

AES command packet size.

#### 20.71.2.2 AES\_DATA\_SIZE

```
#define AES_DATA_SIZE (16)
```

size of AES encrypt/decrypt data

#### 20.71.2.3 AES\_INPUT\_IDX

```
#define AES_INPUT_IDX ATCA_DATA_IDX
```

AES command index for input data.

#### 20.71.2.4 AES\_KEYID\_IDX

```
#define AES_KEYID_IDX ATCA_PARAM2_IDX
```

AES command index for key id.

#### 20.71.2.5 AES\_MODE\_DECRYPT

```
#define AES_MODE_DECRYPT ((uint8_t)0x01)
```

AES mode: Decrypt.



#### 20.71.2.6 AES\_MODE\_ENCRYPT

```
#define AES_MODE_ENCRYPT ((uint8_t)0x00)
```

AES mode: Encrypt.

#### 20.71.2.7 AES\_MODE\_GFM

```
#define AES_MODE_GFM ((uint8_t)0x03)
```

AES mode: GFM calculation.

#### 20.71.2.8 AES\_MODE\_IDX

```
#define AES_MODE_IDX ATCA_PARAM1_IDX
```

AES command index for mode.

#### 20.71.2.9 AES\_MODE\_KEY\_BLOCK\_MASK

```
#define AES_MODE_KEY_BLOCK_MASK ((uint8_t)0xC0)
```

AES mode mask for key block field.

#### 20.71.2.10 AES\_MODE\_KEY\_BLOCK\_POS

```
#define AES_MODE_KEY_BLOCK_POS (6)
```

Bit shift for key block in mode.

#### 20.71.2.11 AES\_MODE\_MASK

```
#define AES_MODE_MASK ((uint8_t)0xC7)
```

AES mode bits 3 to 5 are 0.

### 20.71.2.12 AES\_MODE\_OP\_MASK

```
#define AES_MODE_OP_MASK ((uint8_t)0x07)
```

AES mode operation mask.

### 20.71.2.13 AES\_RSP\_SIZE

```
#define AES_RSP_SIZE ATCA_RSP_SIZE_16
```

AES command response packet size.

### 20.71.2.14 ATCA\_ADDRESS\_MASK

```
#define ATCA_ADDRESS_MASK (0x007F)
```

Address bit 7 to 15 are always 0.

### 20.71.2.15 ATCA\_ADDRESS\_MASK\_CONFIG

```
#define ATCA_ADDRESS_MASK_CONFIG (0x001F)
```

Address bits 5 to 7 are 0 for Configuration zone.

### 20.71.2.16 ATCA\_ADDRESS\_MASK\_OTP

```
#define ATCA_ADDRESS_MASK_OTP (0x000F)
```

Address bits 4 to 7 are 0 for OTP zone.

### 20.71.2.17 ATCA\_AES

```
#define ATCA_AES ((uint8_t)0x51)
```

AES command op-code.

**20.71.2.18 ATCA\_AES\_GFM\_SIZE**

```
#define ATCA_AES_GFM_SIZE ATCA_BLOCK_SIZE
```

size of GFM data

**20.71.2.19 ATCA\_AES\_KEY\_TYPE**

```
#define ATCA_AES_KEY_TYPE 6
```

AES-128 Key.

**20.71.2.20 ATCA\_B283\_KEY\_TYPE**

```
#define ATCA_B283_KEY_TYPE 0
```

B283 NIST ECC key.

**20.71.2.21 ATCA\_BLOCK\_SIZE**

```
#define ATCA_BLOCK_SIZE (32)
```

size of a block

**20.71.2.22 ATCA\_CHECKMAC**

```
#define ATCA_CHECKMAC ((uint8_t)0x28)
```

CheckMac command op-code.

**20.71.2.23 ATCA\_CHIPMODE\_CLOCK\_DIV\_M0**

```
#define ATCA_CHIPMODE_CLOCK_DIV_M0 ((uint8_t)0x00)
```

ChipMode clock divider M0.

### 20.71.2.24 ATCA\_CHIPMODE\_CLOCK\_DIV\_M1

```
#define ATCA_CHIPMODE_CLOCK_DIV_M1 ((uint8_t)0x28)
```

ChipMode clock divider M1.

### 20.71.2.25 ATCA\_CHIPMODE\_CLOCK\_DIV\_M2

```
#define ATCA_CHIPMODE_CLOCK_DIV_M2 ((uint8_t)0x68)
```

ChipMode clock divider M2.

### 20.71.2.26 ATCA\_CHIPMODE\_CLOCK\_DIV\_MASK

```
#define ATCA_CHIPMODE_CLOCK_DIV_MASK ((uint8_t)0xF8)
```

ChipMode clock divider mask.

### 20.71.2.27 ATCA\_CHIPMODE\_I2C\_ADDRESS\_FLAG

```
#define ATCA_CHIPMODE_I2C_ADDRESS_FLAG ((uint8_t)0x01)
```

ChipMode I2C Address in UserExtraAdd flag.

### 20.71.2.28 ATCA\_CHIPMODE\_OFFSET

```
#define ATCA_CHIPMODE_OFFSET (19)
```

ChipMode byte offset within the configuration zone.

### 20.71.2.29 ATCA\_CHIPMODE\_TTL\_ENABLE\_FLAG

```
#define ATCA_CHIPMODE_TTL_ENABLE_FLAG ((uint8_t)0x02)
```

ChipMode TTLenable flag.

**20.71.2.30 ATCA\_CHIPMODE\_WATCHDOG\_LONG**

```
#define ATCA_CHIPMODE_WATCHDOG_LONG ((uint8_t)0x04)
```

ChipMode long watchdog (~13s)

**20.71.2.31 ATCA\_CHIPMODE\_WATCHDOG\_MASK**

```
#define ATCA_CHIPMODE_WATCHDOG_MASK ((uint8_t)0x04)
```

ChipMode watchdog duration mask.

**20.71.2.32 ATCA\_CHIPMODE\_WATCHDOG\_SHORT**

```
#define ATCA_CHIPMODE_WATCHDOG_SHORT ((uint8_t)0x00)
```

ChipMode short watchdog (~1.3s)

**20.71.2.33 ATCA\_CMD\_SIZE\_MAX**

```
#define ATCA_CMD_SIZE_MAX ((uint8_t)4 * 36 + 7)
```

maximum size of command packet (Verify)

**20.71.2.34 ATCA\_CMD\_SIZE\_MIN**

```
#define ATCA_CMD_SIZE_MIN ((uint8_t)7)
```

minimum number of bytes in command (from count byte to second CRC byte)

**20.71.2.35 ATCA\_COUNT\_IDX**

```
#define ATCA_COUNT_IDX (0)
```

command packet index for count

### 20.71.2.36 ATCA\_COUNT\_SIZE

```
#define ATCA_COUNT_SIZE ((uint8_t)1)
```

Number of bytes in the command packet Count.

### 20.71.2.37 ATCA\_COUNTER

```
#define ATCA_COUNTER ((uint8_t)0x24)
```

Counter command op-code.

### 20.71.2.38 ATCA\_CRC\_SIZE

```
#define ATCA_CRC_SIZE ((uint8_t)2)
```

Number of bytes in the command packet CRC.

### 20.71.2.39 ATCA\_DATA\_IDX

```
#define ATCA_DATA_IDX (5)
```

command packet index for data load

### 20.71.2.40 ATCA\_DATA\_SIZE

```
#define ATCA_DATA_SIZE (ATCA_KEY_COUNT * ATCA_KEY_SIZE)
```

size of data zone

### 20.71.2.41 ATCA\_DERIVE\_KEY

```
#define ATCA_DERIVE_KEY ((uint8_t)0x1C)
```

DeriveKey command op-code.

**20.71.2.42 ATCA\_ECC\_CONFIG\_SIZE**

```
#define ATCA_ECC_CONFIG_SIZE (128)
```

size of configuration zone

**20.71.2.43 ATCA\_ECDH**

```
#define ATCA_ECDH ((uint8_t)0x43)
```

ECDH command op-code.

**20.71.2.44 ATCA\_GENDIG**

```
#define ATCA_GENDIG ((uint8_t)0x15)
```

GenDig command op-code.

**20.71.2.45 ATCA\_GENKEY**

```
#define ATCA_GENKEY ((uint8_t)0x40)
```

GenKey command op-code.

**20.71.2.46 ATCA\_HMAC**

```
#define ATCA_HMAC ((uint8_t)0x11)
```

HMAC command op-code.

**20.71.2.47 ATCA\_INFO**

```
#define ATCA_INFO ((uint8_t)0x30)
```

Info command op-code.

### 20.71.2.48 ATCA\_K283\_KEY\_TYPE

```
#define ATCA_K283_KEY_TYPE 1
```

K283 NIST ECC key.

### 20.71.2.49 ATCA\_KDF

```
#define ATCA_KDF ((uint8_t)0x56)
```

KDF command op-code.

### 20.71.2.50 ATCA\_KEY\_COUNT

```
#define ATCA_KEY_COUNT (16)
```

number of keys

### 20.71.2.51 ATCA\_KEY\_ID\_MAX

```
#define ATCA_KEY_ID_MAX ((uint8_t)15)
```

maximum value for key id

### 20.71.2.52 ATCA\_KEY\_SIZE

```
#define ATCA_KEY_SIZE (32)
```

size of a symmetric SHA key

### 20.71.2.53 ATCA\_LOCK

```
#define ATCA_LOCK ((uint8_t)0x17)
```

Lock command op-code.



**20.71.2.54 ATCA\_LOCKED**

```
#define ATCA_LOCKED (0x00)
```

Value indicating a locked zone.

**20.71.2.55 ATCA\_MAC**

```
#define ATCA_MAC ((uint8_t)0x08)
```

MAC command op-code.

**20.71.2.56 ATCA\_NONCE**

```
#define ATCA_NONCE ((uint8_t)0x16)
```

Nonce command op-code.

**20.71.2.57 ATCA\_OPCODE\_IDX**

```
#define ATCA_OPCODE_IDX (1)
```

command packet index for op-code

**20.71.2.58 ATCA\_OTP\_BLOCK\_MAX**

```
#define ATCA_OTP_BLOCK_MAX ((uint8_t)1)
```

maximum value for OTP block

**20.71.2.59 ATCA\_OTP\_SIZE**

```
#define ATCA_OTP_SIZE (64)
```

size of OTP zone

### 20.71.2.60 ATCA\_P256\_KEY\_TYPE

```
#define ATCA_P256_KEY_TYPE 4
```

P256 NIST ECC key.

### 20.71.2.61 ATCA\_PACKET\_OVERHEAD

```
#define ATCA_PACKET_OVERHEAD (ATCA_COUNT_SIZE + ATCA_CRC_SIZE)
```

Number of bytes in the command packet.

### 20.71.2.62 ATCA\_PARAM1\_IDX

```
#define ATCA_PARAM1_IDX (2)
```

command packet index for first parameter

### 20.71.2.63 ATCA\_PARAM2\_IDX

```
#define ATCA_PARAM2_IDX (3)
```

command packet index for second parameter

### 20.71.2.64 ATCA\_PAUSE

```
#define ATCA_PAUSE ((uint8_t)0x01)
```

Pause command op-code.

### 20.71.2.65 ATCA\_PRIV\_KEY\_SIZE

```
#define ATCA_PRIV_KEY_SIZE (32)
```

size of a p256 private key

**20.71.2.66 ATCA\_PRIVWRITE**

```
#define ATCA_PRIVWRITE ((uint8_t)0x46)
```

PrivWrite command op-code.

**20.71.2.67 ATCA\_PUB\_KEY\_PAD**

```
#define ATCA_PUB_KEY_PAD (4)
```

size of the public key pad

**20.71.2.68 ATCA\_PUB\_KEY\_SIZE**

```
#define ATCA_PUB_KEY_SIZE (64)
```

size of a p256 public key

**20.71.2.69 ATCA\_RANDOM**

```
#define ATCA_RANDOM ((uint8_t)0x1B)
```

Random command op-code.

**20.71.2.70 ATCA\_READ**

```
#define ATCA_READ ((uint8_t)0x02)
```

Read command op-code.

**20.71.2.71 ATCA\_RSP\_DATA\_IDX**

```
#define ATCA_RSP_DATA_IDX (1)
```

buffer index of data in response

### 20.71.2.72 ATCA\_RSP\_SIZE\_16

```
#define ATCA_RSP_SIZE_16 ((uint8_t)19)
```

size of response packet containing 16 bytes data

### 20.71.2.73 ATCA\_RSP\_SIZE\_32

```
#define ATCA_RSP_SIZE_32 ((uint8_t)35)
```

size of response packet containing 32 bytes data

### 20.71.2.74 ATCA\_RSP\_SIZE\_4

```
#define ATCA_RSP_SIZE_4 ((uint8_t)7)
```

size of response packet containing 4 bytes data

### 20.71.2.75 ATCA\_RSP\_SIZE\_64

```
#define ATCA_RSP_SIZE_64 ((uint8_t)67)
```

size of response packet containing 64 bytes data

### 20.71.2.76 ATCA\_RSP\_SIZE\_72

```
#define ATCA_RSP_SIZE_72 ((uint8_t)75)
```

size of response packet containing 64 bytes data

### 20.71.2.77 ATCA\_RSP\_SIZE\_MAX

```
#define ATCA_RSP_SIZE_MAX ((uint8_t)75)
```

maximum size of response packet (GenKey and Verify command)

**20.71.2.78 ATCA\_RSP\_SIZE\_MIN**

```
#define ATCA_RSP_SIZE_MIN ((uint8_t)4)
```

minimum number of bytes in response

**20.71.2.79 ATCA\_RSP\_SIZE\_VAL**

```
#define ATCA_RSP_SIZE_VAL ((uint8_t)7)
```

size of response packet containing four bytes of data

**20.71.2.80 ATCA\_SECUREBOOT**

```
#define ATCA_SECUREBOOT ((uint8_t)0x80)
```

Secure Boot command op-code.

**20.71.2.81 ATCA\_SELFTEST**

```
#define ATCA_SELFTEST ((uint8_t)0x77)
```

Self test command op-code.

**20.71.2.82 ATCA\_SERIAL\_NUM\_SIZE**

```
#define ATCA_SERIAL_NUM_SIZE (9)
```

number of bytes in the device serial number

**20.71.2.83 ATCA\_SHA**

```
#define ATCA_SHA ((uint8_t)0x47)
```

SHA command op-code.

### 20.71.2.84 ATCA\_SHA\_CONFIG\_SIZE

```
#define ATCA_SHA_CONFIG_SIZE (88)
```

size of configuration zone

### 20.71.2.85 ATCA\_SHA\_DIGEST\_SIZE

```
#define ATCA_SHA_DIGEST_SIZE (32)
```

### 20.71.2.86 ATCA\_SHA\_KEY\_TYPE

```
#define ATCA_SHA_KEY_TYPE 7
```

SHA key or other data.

### 20.71.2.87 ATCA\_SIG\_SIZE

```
#define ATCA_SIG_SIZE (64)
```

size of a p256 signature

### 20.71.2.88 ATCA\_SIGN

```
#define ATCA_SIGN ((uint8_t)0x41)
```

Sign command op-code.

### 20.71.2.89 ATCA\_TEMPKEY\_KEYID

```
#define ATCA_TEMPKEY_KEYID (0xFFFF)
```

KeyID when referencing TempKey.

**20.71.2.90 ATCA\_UNLOCKED**

```
#define ATCA_UNLOCKED (0x55)
```

Value indicating an unlocked zone.

**20.71.2.91 ATCA\_UPDATE\_EXTRA**

```
#define ATCA_UPDATE_EXTRA ((uint8_t)0x20)
```

UpdateExtra command op-code.

**20.71.2.92 ATCA\_VERIFY**

```
#define ATCA_VERIFY ((uint8_t)0x45)
```

GenKey command op-code.

**20.71.2.93 ATCA\_WORD\_SIZE**

```
#define ATCA_WORD_SIZE (4)
```

size of a word

**20.71.2.94 ATCA\_WRITE**

```
#define ATCA_WRITE ((uint8_t)0x12)
```

Write command op-code.

**20.71.2.95 ATCA\_ZONE\_ENCRYPTED**

```
#define ATCA_ZONE_ENCRYPTED ((uint8_t)0x40)
```

Zone bit 6 set: Write is encrypted with an unlocked data zone.

### 20.71.2.96 ATCA\_ZONE\_MASK

```
#define ATCA_ZONE_MASK ((uint8_t)0x03)
```

Zone mask.

### 20.71.2.97 ATCA\_ZONE\_READWRITE\_32

```
#define ATCA_ZONE_READWRITE_32 ((uint8_t)0x80)
```

Zone bit 7 set: Access 32 bytes, otherwise 4 bytes.

### 20.71.2.98 CHECKMAC\_CLIENT\_CHALLENGE\_IDX

```
#define CHECKMAC_CLIENT_CHALLENGE_IDX ATCA_DATA_IDX
```

CheckMAC command index for client challenge.

### 20.71.2.99 CHECKMAC\_CLIENT\_CHALLENGE\_SIZE

```
#define CHECKMAC_CLIENT_CHALLENGE_SIZE (32)
```

CheckMAC size of client challenge.

### 20.71.2.100 CHECKMAC\_CLIENT\_COMMAND\_SIZE

```
#define CHECKMAC_CLIENT_COMMAND_SIZE (4)
```

CheckMAC size of client command header size inside "other data".

### 20.71.2.101 CHECKMAC\_CLIENT\_RESPONSE\_IDX

```
#define CHECKMAC_CLIENT_RESPONSE_IDX (37)
```

CheckMAC command index for client response.



**20.71.2.102 CHECKMAC\_CLIENT\_RESPONSE\_SIZE**

```
#define CHECKMAC_CLIENT_RESPONSE_SIZE (32)
```

CheckMAC size of client response.

**20.71.2.103 CHECKMAC\_CMD\_MATCH**

```
#define CHECKMAC_CMD_MATCH (0)
```

CheckMAC return value when there is a match.

**20.71.2.104 CHECKMAC\_CMD\_MISMATCH**

```
#define CHECKMAC_CMD_MISMATCH (1)
```

CheckMAC return value when there is a mismatch.

**20.71.2.105 CHECKMAC\_COUNT**

```
#define CHECKMAC_COUNT (84)
```

CheckMAC command packet size.

**20.71.2.106 CHECKMAC\_DATA\_IDX**

```
#define CHECKMAC_DATA_IDX (69)
```

CheckMAC command index for other data.

**20.71.2.107 CHECKMAC\_KEYID\_IDX**

```
#define CHECKMAC_KEYID_IDX ATCA_PARAM2_IDX
```

CheckMAC command index for key identifier.

### 20.71.2.108 CHECKMAC\_MODE\_BLOCK1\_TEMPKEY

```
#define CHECKMAC_MODE_BLOCK1_TEMPKEY ((uint8_t)0x02)
```

CheckMAC mode bit 1: first SHA block from TempKey.

### 20.71.2.109 CHECKMAC\_MODE\_BLOCK2\_TEMPKEY

```
#define CHECKMAC_MODE_BLOCK2_TEMPKEY ((uint8_t)0x01)
```

CheckMAC mode bit 0: second SHA block from TempKey.

### 20.71.2.110 CHECKMAC\_MODE\_CHALLENGE

```
#define CHECKMAC_MODE_CHALLENGE ((uint8_t)0x00)
```

CheckMAC mode 0: first SHA block from key id.

### 20.71.2.111 CHECKMAC\_MODE\_IDX

```
#define CHECKMAC_MODE_IDX ATCA_PARAM1_IDX
```

CheckMAC command index for mode.

### 20.71.2.112 CHECKMAC\_MODE\_INCLUDE\_OTP\_64

```
#define CHECKMAC_MODE_INCLUDE_OTP_64 ((uint8_t)0x20)
```

CheckMAC mode bit 5: include first 64 OTP bits.

### 20.71.2.113 CHECKMAC\_MODE\_MASK

```
#define CHECKMAC_MODE_MASK ((uint8_t)0x27)
```

CheckMAC mode bits 3, 4, 6, and 7 are 0.

**20.71.2.114 CHECKMAC\_MODE\_SOURCE\_FLAG\_MATCH**

```
#define CHECKMAC_MODE_SOURCE_FLAG_MATCH ((uint8_t)0x04)
```

CheckMAC mode bit 2: match TempKey.SourceFlag.

**20.71.2.115 CHECKMAC\_OTHER\_DATA\_SIZE**

```
#define CHECKMAC_OTHER_DATA_SIZE (13)
```

CheckMAC size of "other data".

**20.71.2.116 CHECKMAC\_RSP\_SIZE**

```
#define CHECKMAC_RSP_SIZE ATCA_RSP_SIZE_MIN
```

CheckMAC response packet size.

**20.71.2.117 CMD\_STATUS\_BYTE\_COMM**

```
#define CMD_STATUS_BYTE_COMM ((uint8_t)0xFF)
```

communication error

**20.71.2.118 CMD\_STATUS\_BYTE\_ECC**

```
#define CMD_STATUS_BYTE_ECC ((uint8_t)0x05)
```

command ECC error

**20.71.2.119 CMD\_STATUS\_BYTE\_EXEC**

```
#define CMD_STATUS_BYTE_EXEC ((uint8_t)0x0F)
```

command execution error

### 20.71.2.120 CMD\_STATUS\_BYTE\_PARSE

```
#define CMD_STATUS_BYTE_PARSE ((uint8_t)0x03)
```

command parse error

### 20.71.2.121 CMD\_STATUS\_SUCCESS

```
#define CMD_STATUS_SUCCESS ((uint8_t)0x00)
```

status byte for success

### 20.71.2.122 CMD\_STATUS\_WAKEUP

```
#define CMD_STATUS_WAKEUP ((uint8_t)0x11)
```

status byte after wake-up

### 20.71.2.123 COUNTER\_COUNT

```
#define COUNTER_COUNT ATCA_CMD_SIZE_MIN
```

### 20.71.2.124 COUNTER\_KEYID\_IDX

```
#define COUNTER_KEYID_IDX ATCA_PARAM2_IDX
```

Counter command index for key id.

### 20.71.2.125 COUNTER\_MAX\_VALUE

```
#define COUNTER_MAX_VALUE ((uint32_t)2097151)
```

Counter maximum value of the counter.

**20.71.2.126 COUNTER\_MODE\_IDX**

```
#define COUNTER_MODE_IDX ATCA_PARAM1_IDX
```

Counter command index for mode.

**20.71.2.127 COUNTER\_MODE\_INCREMENT**

```
#define COUNTER_MODE_INCREMENT ((uint8_t)0x01)
```

Counter command mode for incrementing.

**20.71.2.128 COUNTER\_MODE\_MASK**

```
#define COUNTER_MODE_MASK ((uint8_t)0x01)
```

Counter mode bits 1 to 7 are 0.

**20.71.2.129 COUNTER\_MODE\_READ**

```
#define COUNTER_MODE_READ ((uint8_t)0x00)
```

Counter command mode for reading.

**20.71.2.130 COUNTER\_RSP\_SIZE**

```
#define COUNTER_RSP_SIZE ATCA_RSP_SIZE_4
```

Counter command response packet size.

**20.71.2.131 COUNTER\_SIZE**

```
#define COUNTER_SIZE ATCA_RSP_SIZE_MIN
```

Counter size in binary.

### 20.71.2.132 DERIVE\_KEY\_COUNT\_LARGE

```
#define DERIVE_KEY_COUNT_LARGE (39)
```

DeriveKey command packet size with MAC.

### 20.71.2.133 DERIVE\_KEY\_COUNT\_SMALL

```
#define DERIVE_KEY_COUNT_SMALL ATCA_CMD_SIZE_MIN
```

DeriveKey command packet size without MAC.

### 20.71.2.134 DERIVE\_KEY\_MAC\_IDX

```
#define DERIVE_KEY_MAC_IDX ATCA_DATA_IDX
```

DeriveKey command index for optional MAC.

### 20.71.2.135 DERIVE\_KEY\_MAC\_SIZE

```
#define DERIVE_KEY_MAC_SIZE (32)
```

DeriveKey MAC size.

### 20.71.2.136 DERIVE\_KEY\_MODE

```
#define DERIVE_KEY_MODE ((uint8_t)0x04)
```

DeriveKey command mode set to 4 as in datasheet.

### 20.71.2.137 DERIVE\_KEY\_RANDOM\_FLAG

```
#define DERIVE_KEY_RANDOM_FLAG ((uint8_t)4)
```

DeriveKey 1. parameter; has to match TempKey.SourceFlag.

**20.71.2.138 DERIVE\_KEY\_RANDOM\_IDX**

```
#define DERIVE_KEY_RANDOM_IDX ATCA_PARAM1_IDX
```

DeriveKey command index for random bit.

**20.71.2.139 DERIVE\_KEY\_RSP\_SIZE**

```
#define DERIVE_KEY_RSP_SIZE ATCA_RSP_SIZE_MIN
```

DeriveKey response packet size.

**20.71.2.140 DERIVE\_KEY\_TARGETKEY\_IDX**

```
#define DERIVE_KEY_TARGETKEY_IDX ATCA_PARAM2_IDX
```

DeriveKey command index for target slot.

**20.71.2.141 ECDH\_COUNT**

```
#define ECDH_COUNT (ATCA_CMD_SIZE_MIN + ATCA_PUB_KEY_SIZE)
```

**20.71.2.142 ECDH\_KEY\_SIZE**

```
#define ECDH_KEY_SIZE ATCA_BLOCK_SIZE
```

ECDH output data size.

**20.71.2.143 ECDH\_MODE\_COPY\_COMPATIBLE**

```
#define ECDH_MODE_COPY_COMPATIBLE ((uint8_t)0x00)
```

**20.71.2.144 ECDH\_MODE\_COPY\_EEPROM\_SLOT**

```
#define ECDH_MODE_COPY_EEPROM_SLOT ((uint8_t)0x04)
```

### 20.71.2.145 ECDH\_MODE\_COPY\_MASK

```
#define ECDH_MODE_COPY_MASK ((uint8_t)0x0C)
```

### 20.71.2.146 ECDH\_MODE\_COPY\_OUTPUT\_BUFFER

```
#define ECDH_MODE_COPY_OUTPUT_BUFFER ((uint8_t)0x0C)
```

### 20.71.2.147 ECDH\_MODE\_COPY\_TEMP\_KEY

```
#define ECDH_MODE_COPY_TEMP_KEY ((uint8_t)0x08)
```

### 20.71.2.148 ECDH\_MODE\_OUTPUT\_CLEAR

```
#define ECDH_MODE_OUTPUT_CLEAR ((uint8_t)0x00)
```

### 20.71.2.149 ECDH\_MODE\_OUTPUT\_ENC

```
#define ECDH_MODE_OUTPUT_ENC ((uint8_t)0x02)
```

### 20.71.2.150 ECDH\_MODE\_OUTPUT\_MASK

```
#define ECDH_MODE_OUTPUT_MASK ((uint8_t)0x02)
```

### 20.71.2.151 ECDH\_MODE\_SOURCE\_EEPROM\_SLOT

```
#define ECDH_MODE_SOURCE_EEPROM_SLOT ((uint8_t)0x00)
```

### 20.71.2.152 ECDH\_MODE\_SOURCE\_MASK

```
#define ECDH_MODE_SOURCE_MASK ((uint8_t)0x01)
```



**20.71.2.153 ECDH\_MODE\_SOURCE\_TEMPKEY**

```
#define ECDH_MODE_SOURCE_TEMPKEY ((uint8_t)0x01)
```

**20.71.2.154 ECDH\_PREFIX\_MODE**

```
#define ECDH_PREFIX_MODE ((uint8_t)0x00)
```

**20.71.2.155 ECDH\_RSP\_SIZE**

```
#define ECDH_RSP_SIZE ATCA_RSP_SIZE_64
```

ECDH command packet size.

**20.71.2.156 GENDIG\_COUNT**

```
#define GENDIG_COUNT ATCA_CMD_SIZE_MIN
```

GenDig command packet size without "other data".

**20.71.2.157 GENDIG\_DATA\_IDX**

```
#define GENDIG_DATA_IDX ATCA_DATA_IDX
```

GenDig command index for optional data.

**20.71.2.158 GENDIG\_KEYID\_IDX**

```
#define GENDIG_KEYID_IDX ATCA_PARAM2_IDX
```

GenDig command index for key id.

### 20.71.2.159 GENDIG\_RSP\_SIZE

```
#define GENDIG_RSP_SIZE ATCA_RSP_SIZE_MIN
```

GenDig command response packet size.

### 20.71.2.160 GENDIG\_ZONE\_CONFIG

```
#define GENDIG_ZONE_CONFIG ((uint8_t)0)
```

GenDig zone id config. Use KeyID to specify any of the four 256-bit blocks of the Configuration zone.

### 20.71.2.161 GENDIG\_ZONE\_COUNTER

```
#define GENDIG_ZONE_COUNTER ((uint8_t)4)
```

GenDig zone id counter. KeyID specifies the monotonic counter ID to be included in the message generation.

### 20.71.2.162 GENDIG\_ZONE\_DATA

```
#define GENDIG_ZONE_DATA ((uint8_t)2)
```

GenDig zone id data. Use KeyID to specify a slot in the Data zone or a transport key in the hardware array.

### 20.71.2.163 GENDIG\_ZONE\_IDX

```
#define GENDIG_ZONE_IDX ATCA_PARAM1_IDX
```

GenDig command index for zone.

### 20.71.2.164 GENDIG\_ZONE\_KEY\_CONFIG

```
#define GENDIG_ZONE_KEY_CONFIG ((uint8_t)5)
```

GenDig zone id key config. KeyID specifies the slot for which the configuration information is to be included in the message generation.

**20.71.2.165 GENDIG\_ZONE\_OTP**

```
#define GENDIG_ZONE_OTP ((uint8_t)1)
```

GenDig zone id OTP. Use KeyID to specify either the first or second 256-bit block of the OTP zone.

**20.71.2.166 GENDIG\_ZONE\_SHARED\_NONCE**

```
#define GENDIG_ZONE_SHARED_NONCE ((uint8_t)3)
```

GenDig zone id shared nonce. KeyID specifies the location of the input value in the message generation.

**20.71.2.167 GENKEY\_COUNT**

```
#define GENKEY_COUNT ATCA_CMD_SIZE_MIN
```

GenKey command packet size without "other data".

**20.71.2.168 GENKEY\_COUNT\_DATA**

```
#define GENKEY_COUNT_DATA (10)
```

GenKey command packet size with "other data".

**20.71.2.169 GENKEY\_DATA\_IDX**

```
#define GENKEY_DATA_IDX (5)
```

GenKey command index for other data.

**20.71.2.170 GENKEY\_KEYID\_IDX**

```
#define GENKEY_KEYID_IDX ATCA_PARAM2_IDX
```

GenKey command index for key id.

### 20.71.2.171 GENKEY\_MODE\_DIGEST

```
#define GENKEY_MODE_DIGEST ((uint8_t)0x08)
```

GenKey mode: PubKey digest will be created after the public key is calculated.

### 20.71.2.172 GENKEY\_MODE\_IDX

```
#define GENKEY_MODE_IDX ATCA_PARAM1_IDX
```

GenKey command index for mode.

### 20.71.2.173 GENKEY\_MODE\_MASK

```
#define GENKEY_MODE_MASK ((uint8_t)0x1C)
```

GenKey mode bits 0 to 1 and 5 to 7 are 0.

### 20.71.2.174 GENKEY\_MODE\_PRIVATE

```
#define GENKEY_MODE_PRIVATE ((uint8_t)0x04)
```

GenKey mode: private key generation.

### 20.71.2.175 GENKEY\_MODE\_PUBKEY\_DIGEST

```
#define GENKEY_MODE_PUBKEY_DIGEST ((uint8_t)0x10)
```

GenKey mode: Calculate PubKey digest on the public key in KeyId.

### 20.71.2.176 GENKEY\_MODE\_PUBLIC

```
#define GENKEY_MODE_PUBLIC ((uint8_t)0x00)
```

GenKey mode: public key calculation.

**20.71.2.177 GENKEY\_OTHER\_DATA\_SIZE**

```
#define GENKEY_OTHER_DATA_SIZE (3)
```

GenKey size of "other data".

**20.71.2.178 GENKEY\_PRIVATE\_TO\_TEMPKEY**

```
#define GENKEY_PRIVATE_TO_TEMPKEY ((uint16_t)0xFFFF)
```

GenKey Create private key and store to tempkey (608 only)

**20.71.2.179 GENKEY\_RSP\_SIZE\_LONG**

```
#define GENKEY_RSP_SIZE_LONG ATCA_RSP_SIZE_64
```

GenKey response packet size when returning a public key.

**20.71.2.180 GENKEY\_RSP\_SIZE\_SHORT**

```
#define GENKEY_RSP_SIZE_SHORT ATCA_RSP_SIZE_MIN
```

GenKey response packet size in Digest mode.

**20.71.2.181 HMAC\_COUNT**

```
#define HMAC_COUNT ATCA_CMD_SIZE_MIN
```

HMAC command packet size.

**20.71.2.182 HMAC\_DIGEST\_SIZE**

```
#define HMAC_DIGEST_SIZE (32)
```

HMAC size of digest response.

### 20.71.2.183 HMAC\_KEYID\_IDX

```
#define HMAC_KEYID_IDX ATCA_PARAM2_IDX
```

HMAC command index for key id.

### 20.71.2.184 HMAC\_MODE\_FLAG\_FULLSN

```
#define HMAC_MODE_FLAG_FULLSN ((uint8_t)0x40)
```

HMAC mode bit 6: If set, include the 48 bits SN[2:3] and SN[4:7] in the message.; otherwise, the corresponding message bits are set to zero.

### 20.71.2.185 HMAC\_MODE\_FLAG\_OTP64

```
#define HMAC_MODE_FLAG_OTP64 ((uint8_t)0x20)
```

HMAC mode bit 5: Include the first 64 OTP bits (OTP[0] through OTP[7]) in the message.; otherwise, the corresponding message bits are set to zero. If Mode[4] is set, the value of this mode bit is ignored. Not applicable for ATECC508A.

### 20.71.2.186 HMAC\_MODE\_FLAG\_OTP88

```
#define HMAC_MODE_FLAG_OTP88 ((uint8_t)0x10)
```

HMAC mode bit 4: Include the first 88 OTP bits (OTP[0] through OTP[10]) in the message.; otherwise, the corresponding message bits are set to zero. Not applicable for ATECC508A.

### 20.71.2.187 HMAC\_MODE\_FLAG\_TK\_NORAND

```
#define HMAC_MODE_FLAG_TK_NORAND ((uint8_t)0x04)
```

HMAC mode bit 2: The value of this bit must match the value in TempKey.SourceFlag or the command will return an error.

### 20.71.2.188 HMAC\_MODE\_FLAG\_TK\_RAND

```
#define HMAC_MODE_FLAG_TK_RAND ((uint8_t)0x00)
```

HMAC mode bit 2: The value of this bit must match the value in TempKey.SourceFlag or the command will return an error.

**20.71.2.189 HMAC\_MODE\_IDX**

```
#define HMAC_MODE_IDX ATCA_PARAM1_IDX
```

HMAC command index for mode.

**20.71.2.190 HMAC\_MODE\_MASK**

```
#define HMAC_MODE_MASK ((uint8_t)0x74)
```

HMAC mode bits 0, 1, 3, and 7 are 0.

**20.71.2.191 HMAC\_RSP\_SIZE**

```
#define HMAC_RSP_SIZE ATCA_RSP_SIZE_32
```

HMAC command response packet size.

**20.71.2.192 INFO\_COUNT**

```
#define INFO_COUNT ATCA_CMD_SIZE_MIN
```

Info command packet size.

**20.71.2.193 INFO\_DRIVER\_STATE\_MASK**

```
#define INFO_DRIVER_STATE_MASK ((uint8_t)0x02)
```

Info driver state mask.

**20.71.2.194 INFO\_MODE\_GPIO**

```
#define INFO_MODE_GPIO ((uint8_t)0x03)
```

Info mode GPIO.

### 20.71.2.195 INFO\_MODE\_KEY\_VALID

```
#define INFO_MODE_KEY_VALID ((uint8_t)0x01)
```

Info mode KeyValid.

### 20.71.2.196 INFO\_MODE\_MAX

```
#define INFO_MODE_MAX ((uint8_t)0x03)
```

Info mode maximum value.

### 20.71.2.197 INFO\_MODE\_REVISION

```
#define INFO_MODE_REVISION ((uint8_t)0x00)
```

Info mode Revision.

### 20.71.2.198 INFO\_MODE\_STATE

```
#define INFO_MODE_STATE ((uint8_t)0x02)
```

Info mode State.

### 20.71.2.199 INFO\_MODE\_VOL\_KEY\_PERMIT

```
#define INFO_MODE_VOL_KEY_PERMIT ((uint8_t)0x04)
```

Info mode GPIO.

### 20.71.2.200 INFO\_NO\_STATE

```
#define INFO_NO_STATE ((uint8_t)0x00)
```

Info mode is not the state mode.



**20.71.2.201 INFO\_OUTPUT\_STATE\_MASK**

```
#define INFO_OUTPUT_STATE_MASK ((uint8_t)0x01)
```

Info output state mask.

**20.71.2.202 INFO\_PARAM1\_IDX**

```
#define INFO_PARAM1_IDX ATCA_PARAM1_IDX
```

Info command index for 1. parameter.

**20.71.2.203 INFO\_PARAM2\_IDX**

```
#define INFO_PARAM2_IDX ATCA_PARAM2_IDX
```

Info command index for 2. parameter.

**20.71.2.204 INFO\_PARAM2\_LATCH\_CLEAR**

```
#define INFO_PARAM2_LATCH_CLEAR ((uint16_t)0x0000)
```

Info param2 to clear the persistent latch.

**20.71.2.205 INFO\_PARAM2\_LATCH\_SET**

```
#define INFO_PARAM2_LATCH_SET ((uint16_t)0x0001)
```

Info param2 to set the persistent latch.

**20.71.2.206 INFO\_PARAM2\_SET\_LATCH\_STATE**

```
#define INFO_PARAM2_SET_LATCH_STATE ((uint16_t)0x0002)
```

Info param2 to set the persistent latch state.

### 20.71.2.207 INFO\_RSP\_SIZE

```
#define INFO_RSP_SIZE ATCA_RSP_SIZE_VAL
```

Info command response packet size.

### 20.71.2.208 INFO\_SIZE

```
#define INFO_SIZE ((uint8_t)0x04)
```

Info return size.

### 20.71.2.209 KDF\_DETAILS\_AES\_KEY\_LOC\_MASK

```
#define KDF_DETAILS_AES_KEY_LOC_MASK ((uint32_t)0x00000003)
```

KDF details for AES, key location mask.

### 20.71.2.210 KDF\_DETAILS\_HKDF\_MSG\_LOC\_INPUT

```
#define KDF_DETAILS_HKDF_MSG_LOC_INPUT ((uint32_t)0x00000002)
```

KDF details for HKDF, message location in input parameter.

### 20.71.2.211 KDF\_DETAILS\_HKDF\_MSG\_LOC\_IV

```
#define KDF_DETAILS_HKDF_MSG_LOC_IV ((uint32_t)0x00000003)
```

KDF details for HKDF, message location is a special IV function.

### 20.71.2.212 KDF\_DETAILS\_HKDF\_MSG\_LOC\_MASK

```
#define KDF_DETAILS_HKDF_MSG_LOC_MASK ((uint32_t)0x00000003)
```

KDF details for HKDF, message location mask.

**20.71.2.213 KDF\_DETAILS\_HKDF\_MSG\_LOC\_SLOT**

```
#define KDF_DETAILS_HKDF_MSG_LOC_SLOT ((uint32_t)0x00000000)
```

KDF details for HKDF, message location in slot.

**20.71.2.214 KDF\_DETAILS\_HKDF\_MSG\_LOC\_TEMPKEY**

```
#define KDF_DETAILS_HKDF_MSG_LOC_TEMPKEY ((uint32_t)0x00000001)
```

KDF details for HKDF, message location in TempKey.

**20.71.2.215 KDF\_DETAILS\_HKDF\_ZERO\_KEY**

```
#define KDF_DETAILS_HKDF_ZERO_KEY ((uint32_t)0x00000004)
```

KDF details for HKDF, key is 32 bytes of zero.

**20.71.2.216 KDF\_DETAILS\_IDX**

```
#define KDF_DETAILS_IDX ATCA_DATA_IDX
```

KDF command index for details.

**20.71.2.217 KDF\_DETAILS\_PRF\_AEAD\_MASK**

```
#define KDF_DETAILS_PRF_AEAD_MASK ((uint32_t)0x00000600)
```

KDF details for PRF, AEAD processing mask.

**20.71.2.218 KDF\_DETAILS\_PRF\_AEAD\_MODE0**

```
#define KDF_DETAILS_PRF_AEAD_MODE0 ((uint32_t)0x00000000)
```

KDF details for PRF, AEAD no processing.

### 20.71.2.219 KDF\_DETAILS\_PRF\_AEAD\_MODE1

```
#define KDF_DETAILS_PRF_AEAD_MODE1 ((uint32_t)0x00000200)
```

KDF details for PRF, AEAD First 32 go to target, second 32 go to output buffer.

### 20.71.2.220 KDF\_DETAILS\_PRF\_KEY\_LEN\_16

```
#define KDF_DETAILS_PRF_KEY_LEN_16 ((uint32_t)0x00000000)
```

KDF details for PRF, source key length is 16 bytes.

### 20.71.2.221 KDF\_DETAILS\_PRF\_KEY\_LEN\_32

```
#define KDF_DETAILS_PRF_KEY_LEN_32 ((uint32_t)0x00000001)
```

KDF details for PRF, source key length is 32 bytes.

### 20.71.2.222 KDF\_DETAILS\_PRF\_KEY\_LEN\_48

```
#define KDF_DETAILS_PRF_KEY_LEN_48 ((uint32_t)0x00000002)
```

KDF details for PRF, source key length is 48 bytes.

### 20.71.2.223 KDF\_DETAILS\_PRF\_KEY\_LEN\_64

```
#define KDF_DETAILS_PRF_KEY_LEN_64 ((uint32_t)0x00000003)
```

KDF details for PRF, source key length is 64 bytes.

### 20.71.2.224 KDF\_DETAILS\_PRF\_KEY\_LEN\_MASK

```
#define KDF_DETAILS_PRF_KEY_LEN_MASK ((uint32_t)0x00000003)
```

KDF details for PRF, source key length mask.

**20.71.2.225 KDF\_DETAILS\_PRF\_TARGET\_LEN\_32**

```
#define KDF_DETAILS_PRF_TARGET_LEN_32 ((uint32_t)0x00000000)
```

KDF details for PRF, target length is 32 bytes.

**20.71.2.226 KDF\_DETAILS\_PRF\_TARGET\_LEN\_64**

```
#define KDF_DETAILS_PRF_TARGET_LEN_64 ((uint32_t)0x00000100)
```

KDF details for PRF, target length is 64 bytes.

**20.71.2.227 KDF\_DETAILS\_PRF\_TARGET\_LEN\_MASK**

```
#define KDF_DETAILS_PRF_TARGET_LEN_MASK ((uint32_t)0x00000100)
```

KDF details for PRF, target length mask.

**20.71.2.228 KDF\_DETAILS\_SIZE**

```
#define KDF_DETAILS_SIZE 4
```

KDF details (param3) size.

**20.71.2.229 KDF\_KEYID\_IDX**

```
#define KDF_KEYID_IDX ATCA_PARAM2_IDX
```

KDF command index for key id.

**20.71.2.230 KDF\_MESSAGE\_IDX**

```
#define KDF_MESSAGE_IDX (ATCA_DATA_IDX + KDF_DETAILS_SIZE)
```

### 20.71.2.231 KDF\_MODE\_ALG\_AES

```
#define KDF_MODE_ALG_AES ((uint8_t)0x20)
```

KDF mode AES algorithm.

### 20.71.2.232 KDF\_MODE\_ALG\_HKDF

```
#define KDF_MODE_ALG_HKDF ((uint8_t)0x40)
```

KDF mode HKDF algorithm.

### 20.71.2.233 KDF\_MODE\_ALG\_MASK

```
#define KDF_MODE_ALG_MASK ((uint8_t)0x60)
```

KDF mode algorithm mask.

### 20.71.2.234 KDF\_MODE\_ALG\_PRF

```
#define KDF_MODE_ALG_PRF ((uint8_t)0x00)
```

KDF mode PRF algorithm.

### 20.71.2.235 KDF\_MODE\_IDX

```
#define KDF_MODE_IDX ATCA_PARAM1_IDX
```

KDF command index for mode.

### 20.71.2.236 KDF\_MODE\_SOURCE\_ALTKEYBUF

```
#define KDF_MODE_SOURCE_ALTKEYBUF ((uint8_t)0x03)
```

KDF mode source key in alternate key buffer.

**20.71.2.237 KDF\_MODE\_SOURCE\_MASK**

```
#define KDF_MODE_SOURCE_MASK ((uint8_t)0x03)
```

KDF mode source key mask.

**20.71.2.238 KDF\_MODE\_SOURCE\_SLOT**

```
#define KDF_MODE_SOURCE_SLOT ((uint8_t)0x02)
```

KDF mode source key in a slot.

**20.71.2.239 KDF\_MODE\_SOURCE\_TEMPKEY**

```
#define KDF_MODE_SOURCE_TEMPKEY ((uint8_t)0x00)
```

KDF mode source key in TempKey.

**20.71.2.240 KDF\_MODE\_SOURCE\_TEMPKEY\_UP**

```
#define KDF_MODE_SOURCE_TEMPKEY_UP ((uint8_t)0x01)
```

KDF mode source key in upper TempKey.

**20.71.2.241 KDF\_MODE\_TARGET\_ALTKEYBUF**

```
#define KDF_MODE_TARGET_ALTKEYBUF ((uint8_t)0x0C)
```

KDF mode target key in alternate key buffer.

**20.71.2.242 KDF\_MODE\_TARGET\_MASK**

```
#define KDF_MODE_TARGET_MASK ((uint8_t)0x1C)
```

KDF mode target key mask.

### 20.71.2.243 KDF\_MODE\_TARGET\_OUTPUT

```
#define KDF_MODE_TARGET_OUTPUT ((uint8_t)0x10)
```

KDF mode target key in output buffer.

### 20.71.2.244 KDF\_MODE\_TARGET\_OUTPUT\_ENC

```
#define KDF_MODE_TARGET_OUTPUT_ENC ((uint8_t)0x14)
```

KDF mode target key encrypted in output buffer.

### 20.71.2.245 KDF\_MODE\_TARGET\_SLOT

```
#define KDF_MODE_TARGET_SLOT ((uint8_t)0x08)
```

KDF mode target key in slot.

### 20.71.2.246 KDF\_MODE\_TARGET\_TEMPKEY

```
#define KDF_MODE_TARGET_TEMPKEY ((uint8_t)0x00)
```

KDF mode target key in TempKey.

### 20.71.2.247 KDF\_MODE\_TARGET\_TEMPKEY\_UP

```
#define KDF_MODE_TARGET_TEMPKEY_UP ((uint8_t)0x04)
```

KDF mode target key in upper TempKey.

### 20.71.2.248 LOCK\_COUNT

```
#define LOCK_COUNT ATCA_CMD_SIZE_MIN
```

Lock command packet size.



**20.71.2.249 LOCK\_RSP\_SIZE**

```
#define LOCK_RSP_SIZE ATCA_RSP_SIZE_MIN
```

Lock command response packet size.

**20.71.2.250 LOCK\_SUMMARY\_IDX**

```
#define LOCK_SUMMARY_IDX ATCA_PARAM2_IDX
```

Lock command index for summary.

**20.71.2.251 LOCK\_ZONE\_CONFIG**

```
#define LOCK_ZONE_CONFIG ((uint8_t)0x00)
```

Lock zone is Config.

**20.71.2.252 LOCK\_ZONE\_DATA**

```
#define LOCK_ZONE_DATA ((uint8_t)0x01)
```

Lock zone is OTP or Data.

**20.71.2.253 LOCK\_ZONE\_DATA\_SLOT**

```
#define LOCK_ZONE_DATA_SLOT ((uint8_t)0x02)
```

Lock slot of Data.

**20.71.2.254 LOCK\_ZONE\_IDX**

```
#define LOCK_ZONE_IDX ATCA_PARAM1_IDX
```

Lock command index for zone.

### 20.71.2.255 LOCK\_ZONE\_MASK

```
#define LOCK_ZONE_MASK (0xBF)
```

Lock parameter 1 bits 6 are 0.

### 20.71.2.256 LOCK\_ZONE\_NO\_CRC

```
#define LOCK_ZONE_NO_CRC ((uint8_t)0x80)
```

Lock command: Ignore summary.

### 20.71.2.257 MAC\_CHALLENGE\_IDX

```
#define MAC_CHALLENGE_IDX ATCA_DATA_IDX
```

MAC command index for optional challenge.

### 20.71.2.258 MAC\_CHALLENGE\_SIZE

```
#define MAC_CHALLENGE_SIZE (32)
```

MAC size of challenge.

### 20.71.2.259 MAC\_COUNT\_LONG

```
#define MAC_COUNT_LONG (39)
```

MAC command packet size with challenge.

### 20.71.2.260 MAC\_COUNT\_SHORT

```
#define MAC_COUNT_SHORT ATCA_CMD_SIZE_MIN
```

MAC command packet size without challenge.

**20.71.2.261 MAC\_KEYID\_IDX**

```
#define MAC_KEYID_IDX ATCA_PARAM2_IDX
```

MAC command index for key id.

**20.71.2.262 MAC\_MODE\_BLOCK1\_TEMPKEY**

```
#define MAC_MODE_BLOCK1_TEMPKEY ((uint8_t)0x02)
```

MAC mode bit 1: first SHA block from TempKey.

**20.71.2.263 MAC\_MODE\_BLOCK2\_TEMPKEY**

```
#define MAC_MODE_BLOCK2_TEMPKEY ((uint8_t)0x01)
```

MAC mode bit 0: second SHA block from TempKey.

**20.71.2.264 MAC\_MODE\_CHALLENGE**

```
#define MAC_MODE_CHALLENGE ((uint8_t)0x00)
```

MAC mode 0: first SHA block from data slot.

**20.71.2.265 MAC\_MODE\_IDX**

```
#define MAC_MODE_IDX ATCA_PARAM1_IDX
```

MAC command index for mode.

**20.71.2.266 MAC\_MODE\_INCLUDE\_OTP\_64**

```
#define MAC_MODE_INCLUDE_OTP_64 ((uint8_t)0x20)
```

MAC mode bit 5: include first 64 OTP bits.

### 20.71.2.267 MAC\_MODE\_INCLUDE\_OTP\_88

```
#define MAC_MODE_INCLUDE_OTP_88 ((uint8_t)0x10)
```

MAC mode bit 4: include first 88 OTP bits.

### 20.71.2.268 MAC\_MODE\_INCLUDE\_SN

```
#define MAC_MODE_INCLUDE_SN ((uint8_t)0x40)
```

MAC mode bit 6: include serial number.

### 20.71.2.269 MAC\_MODE\_MASK

```
#define MAC_MODE_MASK ((uint8_t)0x77)
```

MAC mode bits 3 and 7 are 0.

### 20.71.2.270 MAC\_MODE\_PASSTHROUGH

```
#define MAC_MODE_PASSTHROUGH ((uint8_t)0x07)
```

MAC mode bit 0-2: pass-through mode.

### 20.71.2.271 MAC\_MODE\_PTNONCE\_TEMPKEY

```
#define MAC_MODE_PTNONCE_TEMPKEY ((uint8_t)0x06)
```

MAC mode bit 0: second SHA block from TempKey.

### 20.71.2.272 MAC\_MODE\_SOURCE\_FLAG\_MATCH

```
#define MAC_MODE_SOURCE_FLAG_MATCH ((uint8_t)0x04)
```

MAC mode bit 2: match TempKey.SourceFlag.

**20.71.2.273 MAC\_RSP\_SIZE**

```
#define MAC_RSP_SIZE ATCA_RSP_SIZE_32
```

MAC command response packet size.

**20.71.2.274 MAC\_SIZE**

```
#define MAC_SIZE (32)
```

MAC size of response.

**20.71.2.275 NONCE\_COUNT\_LONG**

```
#define NONCE_COUNT_LONG (ATCA_CMD_SIZE_MIN + 32)
```

Nonce command packet size for 32 bytes of NumIn.

**20.71.2.276 NONCE\_COUNT\_LONG\_64**

```
#define NONCE_COUNT_LONG_64 (ATCA_CMD_SIZE_MIN + 64)
```

Nonce command packet size for 64 bytes of NumIn.

**20.71.2.277 NONCE\_COUNT\_SHORT**

```
#define NONCE_COUNT_SHORT (ATCA_CMD_SIZE_MIN + 20)
```

Nonce command packet size for 20 bytes of NumIn.

**20.71.2.278 NONCE\_INPUT\_IDX**

```
#define NONCE_INPUT_IDX ATCA_DATA_IDX
```

Nonce command index for input data.

### 20.71.2.279 NONCE\_MODE\_IDX

```
#define NONCE_MODE_IDX ATCA_PARAM1_IDX
```

Nonce command index for mode.

### 20.71.2.280 NONCE\_MODE\_INPUT\_LEN\_32

```
#define NONCE_MODE_INPUT_LEN_32 ((uint8_t)0x00)
```

Nonce mode: input size is 32 bytes.

### 20.71.2.281 NONCE\_MODE\_INPUT\_LEN\_64

```
#define NONCE_MODE_INPUT_LEN_64 ((uint8_t)0x20)
```

Nonce mode: input size is 64 bytes.

### 20.71.2.282 NONCE\_MODE\_INPUT\_LEN\_MASK

```
#define NONCE_MODE_INPUT_LEN_MASK ((uint8_t)0x20)
```

Nonce mode: input size mask.

### 20.71.2.283 NONCE\_MODE\_INVALID

```
#define NONCE_MODE_INVALID ((uint8_t)0x02)
```

Nonce mode 2 is invalid.

### 20.71.2.284 NONCE\_MODE\_MASK

```
#define NONCE_MODE_MASK ((uint8_t)0x03)
```

Nonce mode bits 2 to 7 are 0.

**20.71.2.285 NONCE\_MODE\_NO\_SEED\_UPDATE**

```
#define NONCE_MODE_NO_SEED_UPDATE ((uint8_t)0x01)
```

Nonce mode: do not update seed.

**20.71.2.286 NONCE\_MODE\_PASSTHROUGH**

```
#define NONCE_MODE_PASSTHROUGH ((uint8_t)0x03)
```

Nonce mode: pass-through.

**20.71.2.287 NONCE\_MODE\_SEED\_UPDATE**

```
#define NONCE_MODE_SEED_UPDATE ((uint8_t)0x00)
```

Nonce mode: update seed.

**20.71.2.288 NONCE\_MODE\_TARGET\_ALTKEYBUF**

```
#define NONCE_MODE_TARGET_ALTKEYBUF ((uint8_t)0x80)
```

Nonce mode: target is Alternate Key Buffer.

**20.71.2.289 NONCE\_MODE\_TARGET\_MASK**

```
#define NONCE_MODE_TARGET_MASK ((uint8_t)0xC0)
```

Nonce mode: target mask.

**20.71.2.290 NONCE\_MODE\_TARGET\_MSGDIGBUF**

```
#define NONCE_MODE_TARGET_MSGDIGBUF ((uint8_t)0x40)
```

Nonce mode: target is Message Digest Buffer.

### 20.71.2.291 NONCE\_MODE\_TARGET\_TEMPKEY

```
#define NONCE_MODE_TARGET_TEMPKEY ((uint8_t)0x00)
```

Nonce mode: target is TempKey.

### 20.71.2.292 NONCE\_NUMIN\_SIZE

```
#define NONCE_NUMIN_SIZE (20)
```

Nonce NumIn size for random modes.

### 20.71.2.293 NONCE\_NUMIN\_SIZE\_PASSTHROUGH

```
#define NONCE_NUMIN_SIZE_PASSTHROUGH (32)
```

Nonce NumIn size for 32-byte pass-through mode.

### 20.71.2.294 NONCE\_PARAM2\_IDX

```
#define NONCE_PARAM2_IDX ATCA_PARAM2_IDX
```

Nonce command index for 2. parameter.

### 20.71.2.295 NONCE\_RSP\_SIZE\_LONG

```
#define NONCE_RSP_SIZE_LONG ATCA_RSP_SIZE_32
```

Nonce command response packet size with output.

### 20.71.2.296 NONCE\_RSP\_SIZE\_SHORT

```
#define NONCE_RSP_SIZE_SHORT ATCA_RSP_SIZE_MIN
```

Nonce command response packet size with no output.



**20.71.2.297 NONCE\_ZERO\_CALC\_MASK**

```
#define NONCE_ZERO_CALC_MASK ((uint16_t)0x8000)
```

Nonce zero (param2): calculation mode mask.

**20.71.2.298 NONCE\_ZERO\_CALC\_RANDOM**

```
#define NONCE_ZERO_CALC_RANDOM ((uint16_t)0x0000)
```

Nonce zero (param2): calculation mode random, use RNG in calculation and return RNG output.

**20.71.2.299 NONCE\_ZERO\_CALC\_TEMPKEY**

```
#define NONCE_ZERO_CALC_TEMPKEY ((uint16_t)0x8000)
```

Nonce zero (param2): calculation mode TempKey, use TempKey in calculation and return new TempKey value.

**20.71.2.300 OUTNONCE\_SIZE**

```
#define OUTNONCE_SIZE (32)
```

Size of the OutNonce response expected from several commands.

**20.71.2.301 PAUSE\_COUNT**

```
#define PAUSE_COUNT ATCA_CMD_SIZE_MIN
```

Pause command packet size.

**20.71.2.302 PAUSE\_PARAM2\_IDX**

```
#define PAUSE_PARAM2_IDX ATCA_PARAM2_IDX
```

Pause command index for 2. parameter.

### 20.71.2.303 PAUSE\_RSP\_SIZE

```
#define PAUSE_RSP_SIZE ATCA_RSP_SIZE_MIN
```

Pause command response packet size.

### 20.71.2.304 PAUSE\_SELECT\_IDX

```
#define PAUSE_SELECT_IDX ATCA_PARAM1_IDX
```

Pause command index for Selector.

### 20.71.2.305 PRIVWRITE\_COUNT

```
#define PRIVWRITE_COUNT (75)
```

PrivWrite command packet size.

### 20.71.2.306 PRIVWRITE\_KEYID\_IDX

```
#define PRIVWRITE_KEYID_IDX ATCA_PARAM2_IDX
```

PrivWrite command index for KeyID.

### 20.71.2.307 PRIVWRITE\_MAC\_IDX

```
#define PRIVWRITE_MAC_IDX (41)
```

PrivWrite command index for MAC.

### 20.71.2.308 PRIVWRITE\_MODE\_ENCRYPT

```
#define PRIVWRITE_MODE_ENCRYPT ((uint8_t)0x40)
```

PrivWrite mode: encrypted.

**20.71.2.309 PRIVWRITE\_RSP\_SIZE**

```
#define PRIVWRITE_RSP_SIZE ATCA_RSP_SIZE_MIN
```

PrivWrite command response packet size.

**20.71.2.310 PRIVWRITE\_VALUE\_IDX**

```
#define PRIVWRITE_VALUE_IDX ( 5)
```

PrivWrite command index for value.

**20.71.2.311 PRIVWRITE\_ZONE\_IDX**

```
#define PRIVWRITE_ZONE_IDX ATCA_PARAM1_IDX
```

PrivWrite command index for zone.

**20.71.2.312 PRIVWRITE\_ZONE\_MASK**

```
#define PRIVWRITE_ZONE_MASK ((uint8_t)0x40)
```

PrivWrite zone bits 0 to 5 and 7 are 0.

**20.71.2.313 RANDOM\_COUNT**

```
#define RANDOM_COUNT ATCA_CMD_SIZE_MIN
```

Random command packet size.

**20.71.2.314 RANDOM\_MODE\_IDX**

```
#define RANDOM_MODE_IDX ATCA_PARAM1_IDX
```

Random command index for mode.

### 20.71.2.315 RANDOM\_NO\_SEED\_UPDATE

```
#define RANDOM_NO_SEED_UPDATE ((uint8_t)0x01)
```

Random mode for no seed update.

### 20.71.2.316 RANDOM\_NUM\_SIZE

```
#define RANDOM_NUM_SIZE ((uint8_t)32)
```

Number of bytes in the data packet of a random command.

### 20.71.2.317 RANDOM\_PARAM2\_IDX

```
#define RANDOM_PARAM2_IDX ATCA_PARAM2_IDX
```

Random command index for 2. parameter.

### 20.71.2.318 RANDOM\_RSP\_SIZE

```
#define RANDOM_RSP_SIZE ATCA_RSP_SIZE_32
```

Random command response packet size.

### 20.71.2.319 RANDOM\_SEED\_UPDATE

```
#define RANDOM_SEED_UPDATE ((uint8_t)0x00)
```

Random mode for automatic seed update.

### 20.71.2.320 READ\_32\_RSP\_SIZE

```
#define READ_32_RSP_SIZE ATCA_RSP_SIZE_32
```

Read command response packet size when reading 32 bytes.

**20.71.2.321 READ\_4\_RSP\_SIZE**

```
#define READ_4_RSP_SIZE ATCA_RSP_SIZE_VAL
```

Read command response packet size when reading 4 bytes.

**20.71.2.322 READ\_ADDR\_IDX**

```
#define READ_ADDR_IDX ATCA_PARAM2_IDX
```

Read command index for address.

**20.71.2.323 READ\_COUNT**

```
#define READ_COUNT ATCA_CMD_SIZE_MIN
```

Read command packet size.

**20.71.2.324 READ\_ZONE\_IDX**

```
#define READ_ZONE_IDX ATCA_PARAM1_IDX
```

Read command index for zone.

**20.71.2.325 READ\_ZONE\_MASK**

```
#define READ_ZONE_MASK ((uint8_t) 0x83)
```

Read zone bits 2 to 6 are 0.

**20.71.2.326 RSA2048\_KEY\_SIZE**

```
#define RSA2048_KEY_SIZE (256)
```

size of a RSA private key

### 20.71.2.327 SECUREBOOT\_COUNT\_DIG

```
#define SECUREBOOT_COUNT_DIG (ATCA_CMD_SIZE_MIN + SECUREBOOT_DIGEST_SIZE)
```

SecureBoot command packet size for just a digest.

### 20.71.2.328 SECUREBOOT\_COUNT\_DIG\_SIG

```
#define SECUREBOOT_COUNT_DIG_SIG (ATCA_CMD_SIZE_MIN + SECUREBOOT_DIGEST_SIZE + SECUREBOOT_SIGNATURE_SIZE)
```

SecureBoot command packet size for a digest and signature.

### 20.71.2.329 SECUREBOOT\_DIGEST\_SIZE

```
#define SECUREBOOT_DIGEST_SIZE (32)
```

SecureBoot digest input size.

### 20.71.2.330 SECUREBOOT\_MAC\_SIZE

```
#define SECUREBOOT_MAC_SIZE (32)
```

SecureBoot MAC output size.

### 20.71.2.331 SECUREBOOT\_MODE\_ENC\_MAC\_FLAG

```
#define SECUREBOOT_MODE_ENC_MAC_FLAG ((uint8_t)0x80)
```

SecureBoot mode flag for encrypted digest and returning validating MAC.

### 20.71.2.332 SECUREBOOT\_MODE\_FULL

```
#define SECUREBOOT_MODE_FULL ((uint8_t)0x05)
```

SecureBoot mode Full.

**20.71.2.333 SECUREBOOT\_MODE\_FULL\_COPY**

```
#define SECUREBOOT_MODE_FULL_COPY ((uint8_t)0x07)
```

SecureBoot mode FullCopy.

**20.71.2.334 SECUREBOOT\_MODE\_FULL\_STORE**

```
#define SECUREBOOT_MODE_FULL_STORE ((uint8_t)0x06)
```

SecureBoot mode FullStore.

**20.71.2.335 SECUREBOOT\_MODE\_IDX**

```
#define SECUREBOOT_MODE_IDX ATCA_PARAM1_IDX
```

SecureBoot command index for mode.

**20.71.2.336 SECUREBOOT\_MODE\_MASK**

```
#define SECUREBOOT_MODE_MASK ((uint8_t)0x07)
```

SecureBoot mode mask.

**20.71.2.337 SECUREBOOT\_MODE\_PROHIBIT\_FLAG**

```
#define SECUREBOOT_MODE_PROHIBIT_FLAG ((uint8_t)0x40)
```

SecureBoot mode flag to prohibit SecureBoot until next power cycle.

**20.71.2.338 SECUREBOOT\_RSP\_SIZE\_MAC**

```
#define SECUREBOOT_RSP_SIZE_MAC (ATCA_PACKET_OVERHEAD + SECUREBOOT_MAC_SIZE)
```

SecureBoot response packet size with MAC.

### 20.71.2.339 SECUREBOOT\_RSP\_SIZE\_NO\_MAC

```
#define SECUREBOOT_RSP_SIZE_NO_MAC ATCA_RSP_SIZE_MIN
```

SecureBoot response packet size for no MAC.

### 20.71.2.340 SECUREBOOT\_SIGNATURE\_SIZE

```
#define SECUREBOOT_SIGNATURE_SIZE (64)
```

SecureBoot signature input size.

### 20.71.2.341 SECUREBOOTCONFIG\_MODE\_DISABLED

```
#define SECUREBOOTCONFIG_MODE_DISABLED ((uint16_t)0x0000)
```

Disabled SecureBootMode in SecureBootConfig value.

### 20.71.2.342 SECUREBOOTCONFIG\_MODE\_FULL\_BOTH

```
#define SECUREBOOTCONFIG_MODE_FULL_BOTH ((uint16_t)0x0001)
```

Both digest and signature always required SecureBootMode in SecureBootConfig value.

### 20.71.2.343 SECUREBOOTCONFIG\_MODE\_FULL\_DIG

```
#define SECUREBOOTCONFIG_MODE_FULL_DIG ((uint16_t)0x0003)
```

Digest stored SecureBootMode in SecureBootConfig value.

### 20.71.2.344 SECUREBOOTCONFIG\_MODE\_FULL\_SIG

```
#define SECUREBOOTCONFIG_MODE_FULL_SIG ((uint16_t)0x0002)
```

Signature stored SecureBootMode in SecureBootConfig value.



**20.71.2.345 SECUREBOOTCONFIG\_MODE\_MASK**

```
#define SECUREBOOTCONFIG_MODE_MASK ((uint16_t)0x0003)
```

Mask for SecureBootMode field in SecureBootConfig value.

**20.71.2.346 SECUREBOOTCONFIG\_OFFSET**

```
#define SECUREBOOTCONFIG_OFFSET (70)
```

SecureBootConfig byte offset into the configuration zone.

**20.71.2.347 SELFTEST\_COUNT**

```
#define SELFTEST_COUNT ATCA_CMD_SIZE_MIN
```

SelfTest command packet size.

**20.71.2.348 SELFTEST\_MODE\_AES**

```
#define SELFTEST_MODE_AES ((uint8_t)0x10)
```

SelfTest mode AES encrypt function.

**20.71.2.349 SELFTEST\_MODE\_ALL**

```
#define SELFTEST_MODE_ALL ((uint8_t)0x3B)
```

SelfTest mode all algorithms.

**20.71.2.350 SELFTEST\_MODE\_ECDH**

```
#define SELFTEST_MODE_ECDH ((uint8_t)0x08)
```

SelfTest mode ECDH function.

### 20.71.2.351 SELFTEST\_MODE\_ECDSA\_SIGN\_VERIFY

```
#define SELFTEST_MODE_ECDSA_SIGN_VERIFY ((uint8_t)0x02)
```

SelfTest mode ECDSA verify function.

### 20.71.2.352 SELFTEST\_MODE\_IDX

```
#define SELFTEST_MODE_IDX ATCA_PARAM1_IDX
```

SelfTest command index for mode.

### 20.71.2.353 SELFTEST\_MODE\_RNG

```
#define SELFTEST_MODE_RNG ((uint8_t)0x01)
```

SelfTest mode RNG DRBG function.

### 20.71.2.354 SELFTEST\_MODE\_SHA

```
#define SELFTEST_MODE_SHA ((uint8_t)0x20)
```

SelfTest mode SHA function.

### 20.71.2.355 SELFTEST\_RSP\_SIZE

```
#define SELFTEST_RSP_SIZE ATCA_RSP_SIZE_MIN
```

SelfTest command response packet size.

### 20.71.2.356 SHA\_COUNT\_LONG

```
#define SHA_COUNT_LONG ATCA_CMD_SIZE_MIN
```

Just a starting size.

**20.71.2.357 SHA\_COUNT\_SHORT**

```
#define SHA_COUNT_SHORT ATCA_CMD_SIZE_MIN
```

**20.71.2.358 SHA\_DATA\_MAX**

```
#define SHA_DATA_MAX (64)
```

**20.71.2.359 SHA\_MODE\_608\_HMAC\_END**

```
#define SHA_MODE_608_HMAC_END ((uint8_t)0x02)
```

Complete the HMAC computation and return digest... Different command on 608.

**20.71.2.360 SHA\_MODE\_HMAC\_END**

```
#define SHA_MODE_HMAC_END ((uint8_t)0x05)
```

Complete the HMAC computation and return digest.

**20.71.2.361 SHA\_MODE\_HMAC\_START**

```
#define SHA_MODE_HMAC_START ((uint8_t)0x04)
```

Initialization, HMAC calculation.

**20.71.2.362 SHA\_MODE\_HMAC\_UPDATE**

```
#define SHA_MODE_HMAC_UPDATE ((uint8_t)0x01)
```

Add 64 bytes in the message to the SHA context.

### 20.71.2.363 SHA\_MODE\_MASK

```
#define SHA_MODE_MASK ((uint8_t)0x07)
```

Mask the bit 0-2.

### 20.71.2.364 SHA\_MODE\_READ\_CONTEXT

```
#define SHA_MODE_READ_CONTEXT ((uint8_t)0x06)
```

Read current SHA-256 context out of the device.

### 20.71.2.365 SHA\_MODE\_SHA256\_END

```
#define SHA_MODE_SHA256_END ((uint8_t)0x02)
```

Complete the calculation and return the digest.

### 20.71.2.366 SHA\_MODE\_SHA256\_PUBLIC

```
#define SHA_MODE_SHA256_PUBLIC ((uint8_t)0x03)
```

Add 64 byte ECC public key in the slot to the SHA context.

### 20.71.2.367 SHA\_MODE\_SHA256\_START

```
#define SHA_MODE_SHA256_START ((uint8_t)0x00)
```

Initialization, does not accept a message.

### 20.71.2.368 SHA\_MODE\_SHA256\_UPDATE

```
#define SHA_MODE_SHA256_UPDATE ((uint8_t)0x01)
```

Add 64 bytes in the meesage to the SHA context.

**20.71.2.369 SHA\_MODE\_TARGET\_MASK**

```
#define SHA_MODE_TARGET_MASK ((uint8_t)0xC0)
```

Resulting digest target location mask.

**20.71.2.370 SHA\_MODE\_WRITE\_CONTEXT**

```
#define SHA_MODE_WRITE_CONTEXT ((uint8_t)0x07)
```

Restore a SHA-256 context into the device.

**20.71.2.371 SHA\_RSP\_SIZE**

```
#define SHA_RSP_SIZE ATCA_RSP_SIZE_32
```

SHA command response packet size.

**20.71.2.372 SHA\_RSP\_SIZE\_LONG**

```
#define SHA_RSP_SIZE_LONG ATCA_RSP_SIZE_32
```

SHA command response packet size.

**20.71.2.373 SHA\_RSP\_SIZE\_SHORT**

```
#define SHA_RSP_SIZE_SHORT ATCA_RSP_SIZE_MIN
```

SHA command response packet size only status code.

**20.71.2.374 SIGN\_COUNT**

```
#define SIGN_COUNT ATCA_CMD_SIZE_MIN
```

Sign command packet size.

### 20.71.2.375 SIGN\_KEYID\_IDX

```
#define SIGN_KEYID_IDX ATCA_PARAM2_IDX
```

Sign command index for key id.

### 20.71.2.376 SIGN\_MODE\_EXTERNAL

```
#define SIGN_MODE_EXTERNAL ((uint8_t)0x80)
```

Sign mode bit 7: external.

### 20.71.2.377 SIGN\_MODE\_IDX

```
#define SIGN_MODE_IDX ATCA_PARAM1_IDX
```

Sign command index for mode.

### 20.71.2.378 SIGN\_MODE\_INCLUDE\_SN

```
#define SIGN_MODE_INCLUDE_SN ((uint8_t)0x40)
```

Sign mode bit 6: include serial number.

### 20.71.2.379 SIGN\_MODE\_INTERNAL

```
#define SIGN_MODE_INTERNAL ((uint8_t)0x00)
```

Sign mode 0: internal.

### 20.71.2.380 SIGN\_MODE\_INVALIDATE

```
#define SIGN_MODE_INVALIDATE ((uint8_t)0x01)
```

Sign mode bit 1: Signature will be used for Verify(Invalidate)

**20.71.2.381 SIGN\_MODE\_MASK**

```
#define SIGN_MODE_MASK ((uint8_t)0xE1)
```

Sign mode bits 1 to 4 are 0.

**20.71.2.382 SIGN\_MODE\_SOURCE\_MASK**

```
#define SIGN_MODE_SOURCE_MASK ((uint8_t)0x20)
```

Sign mode message source mask.

**20.71.2.383 SIGN\_MODE\_SOURCE\_MSGDIGBUF**

```
#define SIGN_MODE_SOURCE_MSGDIGBUF ((uint8_t)0x20)
```

Sign mode message source is the Message Digest Buffer.

**20.71.2.384 SIGN\_MODE\_SOURCE\_TEMPKEY**

```
#define SIGN_MODE_SOURCE_TEMPKEY ((uint8_t)0x00)
```

Sign mode message source is TempKey.

**20.71.2.385 SIGN\_RSP\_SIZE**

```
#define SIGN_RSP_SIZE ATCA_RSP_SIZE_MAX
```

Sign command response packet size.

**20.71.2.386 UPDATE\_COUNT**

```
#define UPDATE_COUNT ATCA_CMD_SIZE_MIN
```

UpdateExtra command packet size.

### 20.71.2.387 UPDATE\_MODE\_DEC\_COUNTER

```
#define UPDATE_MODE_DEC_COUNTER ((uint8_t)0x02)
```

UpdateExtra mode: decrement counter.

### 20.71.2.388 UPDATE\_MODE\_IDX

```
#define UPDATE_MODE_IDX ATCA_PARAM1_IDX
```

UpdateExtra command index for mode.

### 20.71.2.389 UPDATE\_MODE\_SELECTOR

```
#define UPDATE_MODE_SELECTOR ((uint8_t)0x01)
```

UpdateExtra mode update Selector (config byte 85)

### 20.71.2.390 UPDATE\_MODE\_USER\_EXTRA

```
#define UPDATE_MODE_USER_EXTRA ((uint8_t)0x00)
```

UpdateExtra mode update UserExtra (config byte 84)

### 20.71.2.391 UPDATE\_MODE\_USER\_EXTRA\_ADD

```
#define UPDATE_MODE_USER_EXTRA_ADD UPDATE_MODE_SELECTOR
```

UpdateExtra mode update UserExtraAdd (config byte 85)

### 20.71.2.392 UPDATE\_RSP\_SIZE

```
#define UPDATE_RSP_SIZE ATCA_RSP_SIZE_MIN
```

UpdateExtra command response packet size.



**20.71.2.393 UPDATE\_VALUE\_IDX**

```
#define UPDATE_VALUE_IDX ATCA_PARAM2_IDX
```

UpdateExtra command index for new value.

**20.71.2.394 VERIFY\_256\_EXTERNAL\_COUNT**

```
#define VERIFY_256_EXTERNAL_COUNT (135)
```

Verify command packet size for 256-bit key in external mode.

**20.71.2.395 VERIFY\_256\_KEY\_SIZE**

```
#define VERIFY_256_KEY_SIZE ( 64)
```

Verify key size for 256-bit key.

**20.71.2.396 VERIFY\_256\_SIGNATURE\_SIZE**

```
#define VERIFY_256_SIGNATURE_SIZE ( 64)
```

Verify signature size for 256-bit key.

**20.71.2.397 VERIFY\_256\_STORED\_COUNT**

```
#define VERIFY_256_STORED_COUNT ( 71)
```

Verify command packet size for 256-bit key in stored mode.

**20.71.2.398 VERIFY\_256\_VALIDATE\_COUNT**

```
#define VERIFY_256_VALIDATE_COUNT ( 90)
```

Verify command packet size for 256-bit key in validate mode.

### 20.71.2.399 VERIFY\_283\_EXTERNAL\_COUNT

```
#define VERIFY_283_EXTERNAL_COUNT (151)
```

Verify command packet size for 283-bit key in external mode.

### 20.71.2.400 VERIFY\_283\_KEY\_SIZE

```
#define VERIFY_283_KEY_SIZE ( 72)
```

Verify key size for 283-bit key.

### 20.71.2.401 VERIFY\_283\_SIGNATURE\_SIZE

```
#define VERIFY_283_SIGNATURE_SIZE ( 72)
```

Verify signature size for 283-bit key.

### 20.71.2.402 VERIFY\_283\_STORED\_COUNT

```
#define VERIFY_283_STORED_COUNT ( 79)
```

Verify command packet size for 283-bit key in stored mode.

### 20.71.2.403 VERIFY\_283\_VALIDATE\_COUNT

```
#define VERIFY_283_VALIDATE_COUNT ( 98)
```

Verify command packet size for 283-bit key in validate mode.

### 20.71.2.404 VERIFY\_DATA\_IDX

```
#define VERIFY_DATA_IDX ( 5)
```

Verify command index for data.

**20.71.2.405 VERIFY\_KEY\_B283**

```
#define VERIFY_KEY_B283 ((uint16_t)0x0000)
```

Verify key type: B283.

**20.71.2.406 VERIFY\_KEY\_K283**

```
#define VERIFY_KEY_K283 ((uint16_t)0x0001)
```

Verify key type: K283.

**20.71.2.407 VERIFY\_KEY\_P256**

```
#define VERIFY_KEY_P256 ((uint16_t)0x0004)
```

Verify key type: P256.

**20.71.2.408 VERIFY\_KEYID\_IDX**

```
#define VERIFY_KEYID_IDX ATCA_PARAM2_IDX
```

Verify command index for key id.

**20.71.2.409 VERIFY\_MODE\_EXTERNAL**

```
#define VERIFY_MODE_EXTERNAL ((uint8_t)0x02)
```

Verify mode: external.

**20.71.2.410 VERIFY\_MODE\_IDX**

```
#define VERIFY_MODE_IDX ATCA_PARAM1_IDX
```

Verify command index for mode.

### 20.71.2.411 VERIFY\_MODE\_INVALIDATE

```
#define VERIFY_MODE_INVALIDATE ((uint8_t)0x07)
```

Verify mode: invalidate.

### 20.71.2.412 VERIFY\_MODE\_MAC\_FLAG

```
#define VERIFY_MODE_MAC_FLAG ((uint8_t)0x80)
```

Verify mode: MAC.

### 20.71.2.413 VERIFY\_MODE\_MASK

```
#define VERIFY_MODE_MASK ((uint8_t)0x07)
```

Verify mode bits 3 to 7 are 0.

### 20.71.2.414 VERIFY\_MODE\_SOURCE\_MASK

```
#define VERIFY_MODE_SOURCE_MASK ((uint8_t)0x20)
```

Verify mode message source mask.

### 20.71.2.415 VERIFY\_MODE\_SOURCE\_MSGDIGBUF

```
#define VERIFY_MODE_SOURCE_MSGDIGBUF ((uint8_t)0x20)
```

Verify mode message source is the Message Digest Buffer.

### 20.71.2.416 VERIFY\_MODE\_SOURCE\_TEMPKEY

```
#define VERIFY_MODE_SOURCE_TEMPKEY ((uint8_t)0x00)
```

Verify mode message source is TempKey.

**20.71.2.417 VERIFY\_MODE\_STORED**

```
#define VERIFY_MODE_STORED ((uint8_t)0x00)
```

Verify mode: stored.

**20.71.2.418 VERIFY\_MODE\_VALIDATE**

```
#define VERIFY_MODE_VALIDATE ((uint8_t)0x03)
```

Verify mode: validate.

**20.71.2.419 VERIFY\_MODE\_VALIDATE\_EXTERNAL**

```
#define VERIFY_MODE_VALIDATE_EXTERNAL ((uint8_t)0x01)
```

Verify mode: validate external.

**20.71.2.420 VERIFY\_OTHER\_DATA\_SIZE**

```
#define VERIFY_OTHER_DATA_SIZE ( 19)
```

Verify size of "other data".

**20.71.2.421 VERIFY\_RSP\_SIZE**

```
#define VERIFY_RSP_SIZE ATCA_RSP_SIZE_MIN
```

Verify command response packet size.

**20.71.2.422 VERIFY\_RSP\_SIZE\_MAC**

```
#define VERIFY_RSP_SIZE_MAC ATCA_RSP_SIZE_32
```

Verify command response packet size with validating MAC.

### 20.71.2.423 WRITE\_ADDR\_IDX

```
#define WRITE_ADDR_IDX ATCA_PARAM2_IDX
```

Write command index for address.

### 20.71.2.424 WRITE\_MAC\_SIZE

```
#define WRITE_MAC_SIZE (32)
```

Write MAC size.

### 20.71.2.425 WRITE\_MAC\_VL\_IDX

```
#define WRITE_MAC_VL_IDX (37)
```

Write command index for MAC following long data.

### 20.71.2.426 WRITE\_MAC\_VS\_IDX

```
#define WRITE_MAC_VS_IDX ( 9)
```

Write command index for MAC following short data.

### 20.71.2.427 WRITE\_RSP\_SIZE

```
#define WRITE_RSP_SIZE ATCA_RSP_SIZE_MIN
```

Write command response packet size.

### 20.71.2.428 WRITE\_VALUE\_IDX

```
#define WRITE_VALUE_IDX ATCA_DATA_IDX
```

Write command index for data.

**20.71.2.429 WRITE\_ZONE\_DATA**

```
#define WRITE_ZONE_DATA ((uint8_t)2)
```

Write zone id data.

**20.71.2.430 WRITE\_ZONE\_IDX**

```
#define WRITE_ZONE_IDX ATCA_PARAM1_IDX
```

Write command index for zone.

**20.71.2.431 WRITE\_ZONE\_MASK**

```
#define WRITE_ZONE_MASK ((uint8_t)0xC3)
```

Write zone bits 2 to 5 are 0.

**20.71.2.432 WRITE\_ZONE\_OTP**

```
#define WRITE_ZONE_OTP ((uint8_t)1)
```

Write zone id OTP.

**20.71.2.433 WRITE\_ZONE\_WITH\_MAC**

```
#define WRITE_ZONE_WITH_MAC ((uint8_t)0x40)
```

Write zone bit 6: write encrypted with MAC.

**20.71.3 Function Documentation****20.71.3.1 atAES()**

```
ATCA_STATUS atAES (  
    ATCACommand ca_cmd,  
    ATCAPacket * packet )
```

ATCACommand AES method.

### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

### Returns

ATCA\_SUCCESS

### 20.71.3.2 atCalcCrc()

```
void atCalcCrc (
    ATCAPacket * packet )
```

This function calculates CRC and adds it to the correct offset in the packet data.

### Parameters

in	<i>packet</i>	Packet to calculate CRC data for
----	---------------	----------------------------------

### 20.71.3.3 atCheckCrc()

```
ATCA_STATUS atCheckCrc (
    const uint8_t * response )
```

This function checks the consistency of a response.

### Parameters

in	<i>response</i>	pointer to response
----	-----------------	---------------------

### Returns

ATCA\_SUCCESS on success, otherwise ATCA\_RX\_CRC\_ERROR

### 20.71.3.4 atCheckMAC()

```
ATCA_STATUS atCheckMAC (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand CheckMAC method.



**Parameters**

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

**Returns**

ATCA\_SUCCESS

**20.71.3.5 atCounter()**

```
ATCA_STATUS atCounter (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Counter method.

**Parameters**

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.71.3.6 atCRC()**

```
void atCRC (
    size_t length,
    const uint8_t * data,
    uint8_t * crc_le )
```

Calculates CRC over the given raw data and returns the CRC in little-endian byte order.

**Parameters**

in	<i>length</i>	Size of data not including the CRC byte positions
in	<i>data</i>	Pointer to the data over which to compute the CRC
out	<i>crc↔ _le</i>	Pointer to the place where the two-bytes of CRC will be returned in little-endian byte order.

### 20.71.3.7 atDeriveKey()

```
ATCA_STATUS atDeriveKey (
    ATCACommand ca_cmd,
    ATCAPacket * packet,
    bool has_mac )
```

ATCACommand DeriveKey method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built
in	<i>has_mac</i>	hasMAC determines if MAC data is present in the packet input

#### Returns

ATCA\_SUCCESS

### 20.71.3.8 atECDH()

```
ATCA_STATUS atECDH (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand ECDH method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

#### Returns

ATCA\_SUCCESS

### 20.71.3.9 atGenDig()

```
ATCA_STATUS atGenDig (
    ATCACommand ca_cmd,
    ATCAPacket * packet,
    bool is_no_mac_key )
```

ATCACommand Generate Digest method.

## Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built
in	<i>is_no_mac_key</i>	Should be true if GenDig is being run on a slot that has its SlotConfig.NoMac bit set

## Returns

ATCA\_SUCCESS

**20.71.3.10 atGenKey()**

```
ATCA_STATUS atGenKey (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Generate Key method.

## Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

## Returns

ATCA\_SUCCESS

**20.71.3.11 atHMAC()**

```
ATCA_STATUS atHMAC (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand HMAC method.

## Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

## Returns

ATCA\_SUCCESS

### 20.71.3.12 atInfo()

```
ATCA_STATUS atInfo (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Info method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

#### Returns

ATCA\_SUCCESS

### 20.71.3.13 atIsECCFamily()

```
bool atIsECCFamily (
    ATCADeviceType device_type )
```

determines if a given device type is an ECC device or a superset of a ECC device

#### Parameters

in	<i>device_type</i>	Type of device to check for family type
----	--------------------	---

#### Returns

boolean indicating whether the given device is an ECC family device.

### 20.71.3.14 atIsSHAFamily()

```
bool atIsSHAFamily (
    ATCADeviceType device_type )
```

determines if a given device type is a SHA device or a superset of a SHA device

#### Parameters

in	<i>device_type</i>	Type of device to check for family type
----	--------------------	---

**Returns**

boolean indicating whether the given device is a SHA family device.

**20.71.3.15 atKDF()**

```
ATCA_STATUS atKDF (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand KDF method.

**Parameters**

in	<i>ca_cmd</i>	Instance
in	<i>packet</i>	Pointer to the packet containing the command being built.

**Returns**

ATCA\_SUCCESS

**20.71.3.16 atLock()**

```
ATCA_STATUS atLock (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Lock method.

**Parameters**

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

**Returns**

ATCA\_SUCCESS

**20.71.3.17 atMAC()**

```
ATCA_STATUS atMAC (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand MAC method.

## Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

## Returns

ATCA\_SUCCESS

**20.71.3.18 atNonce()**

```
ATCA_STATUS atNonce (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Nonce method.

## Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

**20.71.3.19 atPause()**

```
ATCA_STATUS atPause (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Pause method.

## Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

## Returns

ATCA\_SUCCESS

### 20.71.3.20 atPrivWrite()

```
ATCA_STATUS atPrivWrite (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand PrivWrite method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

#### Returns

ATCA\_SUCCESS

### 20.71.3.21 atRandom()

```
ATCA_STATUS atRandom (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Random method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

#### Returns

ATCA\_SUCCESS

### 20.71.3.22 atRead()

```
ATCA_STATUS atRead (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Read method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

### Returns

ATCA\_SUCCESS

### 20.71.3.23 atSecureBoot()

```
ATCA_STATUS atSecureBoot (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand SecureBoot method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

### Returns

ATCA\_SUCCESS

### 20.71.3.24 atSelfTest()

```
ATCA_STATUS atSelfTest (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand AES method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

### Returns

ATCA\_SUCCESS

### 20.71.3.25 atSHA()

```
ATCA_STATUS atSHA (
    ATCACommand ca_cmd,
```



```
ATCAPacket * packet,  
uint16_t write_context_size )
```

ATCACCommand SHA method.

### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built
in	<i>write_context_size</i>	the length of the sha write_context data

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.71.3.26 atSign()

```
ATCA_STATUS atSign (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand Sign method.

### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

### Returns

ATCA\_SUCCESS

### 20.71.3.27 atUpdateExtra()

```
ATCA_STATUS atUpdateExtra (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand UpdateExtra method.

### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

### Returns

ATCA\_SUCCESS

### 20.71.3.28 atVerify()

```
ATCA_STATUS atVerify (
    ATCACommand ca_cmd,
    ATCAPacket * packet )
```

ATCACommand ECDSA Verify method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.71.3.29 atWrite()

```
ATCA_STATUS atWrite (
    ATCACommand ca_cmd,
    ATCAPacket * packet,
    bool has_mac )
```

ATCACommand Write method.

#### Parameters

in	<i>ca_cmd</i>	instance
in	<i>packet</i>	pointer to the packet containing the command being built
in	<i>has_mac</i>	Flag to indicate whether a mac is present or not

#### Returns

ATCA\_SUCCESS

### 20.71.3.30 isATCAError()

```
ATCA_STATUS isATCAError (
    uint8_t * data )
```

checks for basic error frame in data

## 20.72 calib\_counter.c File Reference

---

### Parameters

in	data	pointer to received data - expected to be in the form of a CA device response frame
----	------	---

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.72 calib\_counter.c File Reference

CryptoAuthLib Basic API methods for Counter command.

```
#include "cryptoauthlib.h"
```

### Functions

- [ATCA\\_STATUS calib\\_counter](#) (ATCADevice device, uint8\_t mode, uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Compute the Counter functions.*
- [ATCA\\_STATUS calib\\_counter\\_increment](#) (ATCADevice device, uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Increments one of the device's monotonic counters.*
- [ATCA\\_STATUS calib\\_counter\\_read](#) (ATCADevice device, uint16\_t counter\_id, uint32\_t \*counter\_value)  
*Read one of the device's monotonic counters.*

### 20.72.1 Detailed Description

CryptoAuthLib Basic API methods for Counter command.

The Counter command reads or increments the binary count value for one of the two monotonic counters

#### Note

List of devices that support this command - ATECC508A and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.73 calib\_derivekey.c File Reference

CryptoAuthLib Basic API methods for DeriveKey command.

```
#include "cryptoauthlib.h"
```

## Functions

- [ATCA\\_STATUS calib\\_derivekey](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t target\_key, const uint8\_t \*mac)  
*Executes the DeviveKey command for deriving a new key from a nonce (TempKey) and an existing key.*

### 20.73.1 Detailed Description

CryptoAuthLib Basic API methods for DeriveKey command.

The DeriveKey command combines the current value of a key with the nonce stored in TempKey using SHA-256 and derives a new key.

#### Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.74 calib\_ecdh.c File Reference

CryptoAuthLib Basic API methods for ECDH command.

```
#include "cryptoauthlib.h"
#include "host/atca_host.h"
```

## Functions

- [ATCA\\_STATUS calib\\_ecdh\\_base](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, uint8\_t \*out\_nonce)  
*Base function for generating premaster secret key using ECDH.*
- [ATCA\\_STATUS calib\\_ecdh](#) ([ATCADevice](#) device, uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms)  
*ECDH command with a private key in a slot and the premaster secret is returned in the clear.*
- [ATCA\\_STATUS calib\\_ecdh\\_enc](#) ([ATCADevice](#) device, uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*read\_key, uint16\_t read\_key\_id, const uint8\_t num\_in[[NONCE\\_NUMIN\\_SIZE](#)])  
*ECDH command with a private key in a slot and the premaster secret is read from the next slot.*
- [ATCA\\_STATUS calib\\_ecdh\\_ioenc](#) ([ATCADevice](#) device, uint16\_t key\_id, const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*io\_key)  
*ECDH command with a private key in a slot and the premaster secret is returned encrypted using the IO protection key.*
- [ATCA\\_STATUS calib\\_ecdh\\_tempkey](#) ([ATCADevice](#) device, const uint8\_t \*public\_key, uint8\_t \*pms)  
*ECDH command with a private key in TempKey and the premaster secret is returned in the clear.*
- [ATCA\\_STATUS calib\\_ecdh\\_tempkey\\_ioenc](#) ([ATCADevice](#) device, const uint8\_t \*public\_key, uint8\_t \*pms, const uint8\_t \*io\_key)  
*ECDH command with a private key in TempKey and the premaster secret is returned encrypted using the IO protection key.*

## 20.74.1 Detailed Description

CryptoAuthLib Basic API methods for ECDH command.

The ECDH command implements the Elliptic Curve Diffie-Hellman algorithm to combine an internal private key with an external public key to calculate a shared secret.

### Note

List of devices that support this command - ATECC508A, ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.74.2 Function Documentation

### 20.74.2.1 calib\_ecdh\_enc()

```
ATCA_STATUS calib_ecdh_enc (
    ATCADevice device,
    uint16_t key_id,
    const uint8_t * public_key,
    uint8_t * pms,
    const uint8_t * read_key,
    uint16_t read_key_id,
    const uint8_t num_in[NONCE_NUMIN_SIZE] )
```

ECDH command with a private key in a slot and the premaster secret is read from the next slot.

This function only works for even numbered slots with the proper configuration.

### Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Slot of key for ECDH computation
in	<i>public_key</i>	Public key input to ECDH calculation. X and Y integers in big-endian format. 64 bytes for P256 key.
out	<i>pms</i>	Computed ECDH premaster secret is returned here (32 bytes).
in	<i>read_key</i>	Read key for the premaster secret slot ( <i>key_id</i>  1).
in	<i>read_key_id</i>	Read key slot for <i>read_key</i> .
in	<i>num_in</i>	20 byte host nonce to inject into Nonce calculation

## Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.75 calib\_execution.c File Reference

Implements an execution handler that executes a given command on a device and returns the results.

```
#include "atca_config.h"
#include <stdlib.h>
#include <string.h>
#include "atca_command.h"
#include "atca_device.h"
#include "calib_execution.h"
#include "atca_devtypes.h"
#include "hal/atca_hal.h"
```

## Functions

- [ATCA\\_STATUS calib\\_execute\\_command](#) ([ATCAPacket](#) \*packet, [ATCADevice](#) device)

*Wakes up device, sends the packet, waits for command completion, receives response, and puts the device into the idle state.*

### 20.75.1 Detailed Description

Implements an execution handler that executes a given command on a device and returns the results.

This implementation wraps Polling and No polling (simple wait) schemes into a single method and use it across the library. Polling is used by default, however, by defining the ATCA\_NO\_POLL symbol the code will instead wait an estimated max execution time before requesting the result.

## Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.75.2 Function Documentation

#### 20.75.2.1 calib\_execute\_command()

```
ATCA_STATUS calib_execute_command (
    ATCAPacket * packet,
    ATCADevice device )
```

Wakes up device, sends the packet, waits for command completion, receives response, and puts the device into the idle state.

### Parameters

<code>in, out</code>	<i>packet</i>	As input, the packet to be sent. As output, the data buffer in the packet structure will contain the response.
<code>in</code>	<i>device</i>	CryptoAuthentication device to send the command to.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.76 calib\_execution.h File Reference

Defines an execution handler that executes a given command on a device and returns the results.

```
#include "atca_status.h"
#include "calib_command.h"
#include "atca_device.h"
#include "atca_config.h"
```

### Macros

- `#define ATCA_UNSUPPORTED_CMD ((uint16_t)0xFFFF)`

### Functions

- `ATCA_STATUS calib_execute_command (ATCAPacket *packet, ATCADevice device)`  
*Wakes up device, sends the packet, waits for command completion, receives response, and puts the device into the idle state.*

### 20.76.1 Detailed Description

Defines an execution handler that executes a given command on a device and returns the results.

The basic flow is to wake the device, send the command, wait/poll for completion, and finally receives the response from the device and does basic checks before returning to caller.

This handler supports the ATSHA and ATECC device family.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.76.2 Macro Definition Documentation



### 20.76.2.1 ATCA\_UNSUPPORTED\_CMD

```
#define ATCA_UNSUPPORTED_CMD ((uint16_t)0xFFFF)
```

## 20.76.3 Function Documentation

### 20.76.3.1 calib\_execute\_command()

```
ATCA_STATUS calib_execute_command (
    ATCAPacket * packet,
    ATCADevice device )
```

Wakes up device, sends the packet, waits for command completion, receives response, and puts the device into the idle state.

#### Parameters

in, out	<i>packet</i>	As input, the packet to be sent. As output, the data buffer in the packet structure will contain the response.
in	<i>device</i>	CryptoAuthentication device to send the command to.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.77 calib\_gendig.c File Reference

CryptoAuthLib Basic API methods for GenDig command.

```
#include "cryptoauthlib.h"
```

### Functions

- [ATCA\\_STATUS calib\\_gendig](#) ([ATCADevice](#) device, uint8\_t zone, uint16\_t key\_id, const uint8\_t \*other\_data, uint8\_t other\_data\_size)

*Issues a GenDig command, which performs a SHA256 hash on the source data indicated by zone with the contents of TempKey. See the CryptoAuth datasheet for your chip to see what the values of zone correspond to.*

### 20.77.1 Detailed Description

CryptoAuthLib Basic API methods for GenDig command.

The GenDig command uses SHA-256 to combine a stored value with the contents of TempKey, which must have been valid prior to the execution of this command.

### Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.78 calib\_genkey.c File Reference

CryptoAuthLib Basic API methods for GenKey command.

```
#include "cryptoauthlib.h"
```

### Functions

- [ATCA\\_STATUS calib\\_genkey\\_base](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*other\_data, uint8\_t \*public\_key)  
*Issues GenKey command, which can generate a private key, compute a public key, and/or compute a digest of a public key.*
- [ATCA\\_STATUS calib\\_genkey](#) ([ATCADevice](#) device, uint16\_t key\_id, uint8\_t \*public\_key)  
*Issues GenKey command, which generates a new random private key in slot and returns the public key.*
- [ATCA\\_STATUS calib\\_get\\_pubkey](#) ([ATCADevice](#) device, uint16\_t key\_id, uint8\_t \*public\_key)  
*Uses GenKey command to calculate the public key from an existing private key in a slot.*

### 20.78.1 Detailed Description

CryptoAuthLib Basic API methods for GenKey command.

The GenKey command is used for creating ECC private keys, generating ECC public keys, and for digest calculations involving public keys.

### Note

List of devices that support this command - ATECC108A, ATECC508A, ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.79 calib\_hmac.c File Reference

CryptoAuthLib Basic API methods for HMAC command.

```
#include "cryptoauthlib.h"
```

## Functions

- [ATCA\\_STATUS calib\\_hmac](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t key\_id, uint8\_t \*digest)  
*Issues a HMAC command, which computes an HMAC/SHA-256 digest of a key stored in the device, a challenge, and other information on the device.*

### 20.79.1 Detailed Description

CryptoAuthLib Basic API methods for HMAC command.

The HMAC command computes an HMAC/SHA-256 digest using a key stored in the device over a challenge stored in the TempKey register, and/or other information stored within the device.

#### Note

List of devices that support this command - ATSHA204A, ATECC108A, and ATECC508A . There are differences in the modes that they support. Refer to device datasheets for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.80 calib\_info.c File Reference

CryptoAuthLib Basic API methods for Info command.

```
#include "cryptoauthlib.h"
```

## Functions

- [ATCA\\_STATUS calib\\_info\\_base](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t param2, uint8\_t \*out\_data)  
*Issues an Info command, which return internal device information and can control GPIO and the persistent latch.*
- [ATCA\\_STATUS calib\\_info](#) ([ATCADevice](#) device, uint8\_t \*revision)  
*Use the Info command to get the device revision (DevRev).*
- [ATCA\\_STATUS calib\\_info\\_get\\_latch](#) ([ATCADevice](#) device, bool \*state)  
*Use the Info command to get the persistent latch current state for an ATECC608 device.*
- [ATCA\\_STATUS calib\\_info\\_set\\_latch](#) ([ATCADevice](#) device, bool state)  
*Use the Info command to set the persistent latch state for an ATECC608 device.*

### 20.80.1 Detailed Description

CryptoAuthLib Basic API methods for Info command.

Info command returns a variety of static and dynamic information about the device and its state. Also is used to control the GPIO pin and the persistent latch.

#### Note

The ATSHA204A refers to this command as DevRev instead of Info, however, the OpCode and operation is the same.

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A & ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.81 calib\_kdf.c File Reference

CryptoAuthLib Basic API methods for KDF command.

```
#include "cryptoauthlib.h"
```

### Functions

- [ATCA\\_STATUS calib\\_kdf](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t key\_id, const uint32\_t details, const uint8\_t \*message, uint8\_t \*out\_data, uint8\_t \*out\_nonce)

*Executes the KDF command, which derives a new key in PRF, AES, or HKDF modes.*

### 20.81.1 Detailed Description

CryptoAuthLib Basic API methods for KDF command.

The KDF command implements one of a number of Key Derivation Functions (KDF). Generally this function combines a source key with an input string and creates a result key/digest/array. Three algorithms are currently supported: PRF, HKDF and AES.

#### Note

List of devices that support this command - ATECC608A/B. Refer to device datasheet for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.82 calib\_lock.c File Reference

CryptoAuthLib Basic API methods for Lock command.

```
#include "cryptoauthlib.h"
```

### Functions

- [ATCA\\_STATUS calib\\_lock](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t summary\_crc)  
*The Lock command prevents future modifications of the Configuration and/or Data and OTP zones. If the device is so configured, then this command can be used to lock individual data slots. This command fails if the designated area is already locked.*
- [ATCA\\_STATUS calib\\_lock\\_config\\_zone](#) ([ATCADevice](#) device)  
*Unconditionally (no CRC required) lock the config zone.*
- [ATCA\\_STATUS calib\\_lock\\_config\\_zone\\_crc](#) ([ATCADevice](#) device, uint16\_t summary\_crc)  
*Lock the config zone with summary CRC.*
- [ATCA\\_STATUS calib\\_lock\\_data\\_zone](#) ([ATCADevice](#) device)  
*Unconditionally (no CRC required) lock the data zone (slots and OTP).*
- [ATCA\\_STATUS calib\\_lock\\_data\\_zone\\_crc](#) ([ATCADevice](#) device, uint16\_t summary\_crc)  
*Lock the data zone (slots and OTP) with summary CRC.*
- [ATCA\\_STATUS calib\\_lock\\_data\\_slot](#) ([ATCADevice](#) device, uint16\_t slot)  
*Lock an individual slot in the data zone on an ATECC device. Not available for ATSHA devices. Slot must be configured to be slot lockable (KeyConfig.Lockable=1).*

### 20.82.1 Detailed Description

CryptoAuthLib Basic API methods for Lock command.

The Lock command prevents future modifications of the Configuration zone, enables configured policies for Data and OTP zones, and can render individual slots read-only regardless of configuration.

#### Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.83 calib\_mac.c File Reference

CryptoAuthLib Basic API methods for MAC command.

```
#include "cryptoauthlib.h"
```

### Functions

- [ATCA\\_STATUS calib\\_mac](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*challenge, uint8\_t \*digest)

*Executes MAC command, which computes a SHA-256 digest of a key stored in the device, a challenge, and other information on the device.*

### 20.83.1 Detailed Description

CryptoAuthLib Basic API methods for MAC command.

The MAC command computes a SHA-256 digest of a key stored in the device, a challenge, and other information on the device. The output of this command is the digest of this message.

#### Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.84 calib\_nonce.c File Reference

CryptoAuthLib Basic API methods for Nonce command.

```
#include "cryptoauthlib.h"
```

### Functions

- [ATCA\\_STATUS calib\\_nonce\\_base](#) (ATCADevice device, uint8\_t mode, uint16\_t zero, const uint8\_t \*num\_in, uint8\_t \*rand\_out)  
*Executes Nonce command, which loads a random or fixed nonce/data into the device for use by subsequent commands.*
- [ATCA\\_STATUS calib\\_nonce](#) (ATCADevice device, const uint8\_t \*num\_in)  
*Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.*
- [ATCA\\_STATUS calib\\_nonce\\_load](#) (ATCADevice device, uint8\_t target, const uint8\_t \*num\_in, uint16\_t num\_in\_size)  
*Execute a Nonce command in pass-through mode to load one of the device's internal buffers with a fixed value.*
- [ATCA\\_STATUS calib\\_nonce\\_rand](#) (ATCADevice device, const uint8\_t \*num\_in, uint8\_t \*rand\_out)  
*Execute a Nonce command to generate a random nonce combining a host nonce (num\_in) and a device random number.*
- [ATCA\\_STATUS calib\\_challenge](#) (ATCADevice device, const uint8\_t \*num\_in)  
*Execute a Nonce command in pass-through mode to initialize TempKey to a specified value.*
- [ATCA\\_STATUS calib\\_challenge\\_seed\\_update](#) (ATCADevice device, const uint8\_t \*num\_in, uint8\_t \*rand\_out)  
*Execute a Nonce command to generate a random challenge combining a host nonce (num\_in) and a device random number.*

### 20.84.1 Detailed Description

CryptoAuthLib Basic API methods for Nonce command.

The Nonce command generates a nonce for use by a subsequent commands of the device by combining an internally generated random number with an input value from the system.

#### Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.85 calib\_privwrite.c File Reference

CryptoAuthLib Basic API methods for PrivWrite command.

```
#include "cryptoauthlib.h"  
#include "host/atca_host.h"
```

## Functions

- [ATCA\\_STATUS calib\\_priv\\_write](#) ([ATCADevice](#) device, uint16\_t key\_id, const uint8\_t priv\_key[36], uint16\_t write\_key\_id, const uint8\_t write\_key[32], const uint8\_t num\_in[[NONCE\\_NUMIN\\_SIZE](#)])

*Executes PrivWrite command, to write externally generated ECC private keys into the device.*

### 20.85.1 Detailed Description

CryptoAuthLib Basic API methods for PrivWrite command.

The PrivWrite command is used to write externally generated ECC private keys into the device.

#### Note

List of devices that support this command - ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.85.2 Function Documentation

#### 20.85.2.1 calib\_priv\_write()

```
ATCA_STATUS calib_priv_write (
    ATCADevice device,
    uint16_t key_id,
    const uint8_t priv_key[36],
    uint16_t write_key_id,
    const uint8_t write_key[32],
    const uint8_t num_in[NONCE_NUMIN_SIZE] )
```

Executes PrivWrite command, to write externally generated ECC private keys into the device.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Slot to write the external private key into.
in	<i>priv_key</i>	External private key (36 bytes) to be written. The first 4 bytes should be zero for P256 curve.
in	<i>write_key_id</i>	Write key slot. Ignored if write_key is NULL.
in	<i>write_key</i>	Write key (32 bytes). If NULL, perform an unencrypted PrivWrite, which is only available when the data zone is unlocked.
in	<i>num_in</i>	20 byte host nonce to inject into Nonce calculation

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.86 calib\_random.c File Reference

CryptoAuthLib Basic API methods for Random command.

```
#include "cryptoauthlib.h"
```

### Functions

- [ATCA\\_STATUS calib\\_random](#) (ATCADevice device, uint8\_t \*rand\_out)

*Executes Random command, which generates a 32 byte random number from the CryptoAuth device.*

### 20.86.1 Detailed Description

CryptoAuthLib Basic API methods for Random command.

The Random command generates a random number for use by the system.

### Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.87 calib\_read.c File Reference

CryptoAuthLib Basic API methods for Read command.

```
#include "cryptoauthlib.h"  
#include "host/atca_host.h"
```



## Functions

- **ATCA\_STATUS calib\_read\_zone** (ATCADevice device, uint8\_t zone, uint16\_t slot, uint8\_t block, uint8\_t offset, uint8\_t \*data, uint8\_t len)  
*Executes Read command, which reads either 4 or 32 bytes of data from a given slot, configuration zone, or the OTP zone.*
- **ATCA\_STATUS calib\_read\_serial\_number** (ATCADevice device, uint8\_t \*serial\_number)  
*Executes Read command, which reads the 9 byte serial number of the device from the config zone.*
- **ATCA\_STATUS calib\_is\_slot\_locked** (ATCADevice device, uint16\_t slot, bool \*is\_locked)  
*Executes Read command, which reads the configuration zone to see if the specified slot is locked.*
- **ATCA\_STATUS calib\_is\_locked** (ATCADevice device, uint8\_t zone, bool \*is\_locked)  
*Executes Read command, which reads the configuration zone to see if the specified zone is locked.*
- **ATCA\_STATUS calib\_read\_enc** (ATCADevice device, uint16\_t key\_id, uint8\_t block, uint8\_t \*data, const uint8\_t \*enc\_key, const uint16\_t enc\_key\_id, const uint8\_t num\_in[NONCE\_NUMIN\_SIZE])  
*Executes Read command on a slot configured for encrypted reads and decrypts the data to return it as plaintext.*
- **ATCA\_STATUS calib\_read\_config\_zone** (ATCADevice device, uint8\_t \*config\_data)  
*Executes Read command to read the complete device configuration zone.*
- **ATCA\_STATUS calib\_cmp\_config\_zone** (ATCADevice device, uint8\_t \*config\_data, bool \*same\_config)  
*Compares a specified configuration zone with the configuration zone currently on the device.*
- **ATCA\_STATUS calib\_read\_sig** (ATCADevice device, uint16\_t slot, uint8\_t \*sig)  
*Executes Read command to read a 64 byte ECDSA P256 signature from a slot configured for clear reads.*
- **ATCA\_STATUS calib\_read\_pubkey** (ATCADevice device, uint16\_t slot, uint8\_t \*public\_key)  
*Executes Read command to read an ECC P256 public key from a slot configured for clear reads.*
- **ATCA\_STATUS calib\_read\_bytes\_zone** (ATCADevice device, uint8\_t zone, uint16\_t slot, size\_t offset, uint8\_t \*data, size\_t length)  
*Used to read an arbitrary number of bytes from any zone configured for clear reads.*

### 20.87.1 Detailed Description

CryptoAuthLib Basic API methods for Read command.

The Read command reads words either 4-byte words or 32-byte blocks from one of the memory zones of the device. The data may optionally be encrypted before being returned to the system.

#### Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.87.2 Function Documentation

### 20.87.2.1 calib\_read\_enc()

```
ATCA_STATUS calib_read_enc (
    ATCADevice device,
    uint16_t key_id,
    uint8_t block,
    uint8_t * data,
    const uint8_t * enc_key,
    const uint16_t enc_key_id,
    const uint8_t num_in[NONCE_NUMIN_SIZE] )
```

Executes Read command on a slot configured for encrypted reads and decrypts the data to return it as plaintext.

Data zone must be locked for this command to succeed. Can only read 32 byte blocks.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	The slot ID to read from.
in	<i>block</i>	Index of the 32 byte block within the slot to read.
out	<i>data</i>	Decrypted (plaintext) data from the read is returned here (32 bytes).
in	<i>enc_key</i>	32 byte ReadKey for the slot being read.
in	<i>enc_key_id</i>	KeyID of the ReadKey being used.
in	<i>num_in</i>	20 byte host nonce to inject into Nonce calculation

returns ATCA\_SUCCESS on success, otherwise an error code.

## 20.88 calib\_secureboot.c File Reference

CryptoAuthLib Basic API methods for SecureBoot command.

```
#include "cryptoauthlib.h"
#include "host/atca_host.h"
```

### Functions

- **ATCA\_STATUS calib\_secureboot** (ATCADevice device, uint8\_t mode, uint16\_t param2, const uint8\_t \*digest, const uint8\_t \*signature, uint8\_t \*mac)

*Executes Secure Boot command, which provides support for secure boot of an external MCU or MPU.*

- **ATCA\_STATUS calib\_secureboot\_mac** (ATCADevice device, uint8\_t mode, const uint8\_t \*digest, const uint8\_t \*signature, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)

*Executes Secure Boot command with encrypted digest and validated MAC response using the IO protection key.*

### 20.88.1 Detailed Description

CryptoAuthLib Basic API methods for SecureBoot command.

The SecureBoot command provides support for secure boot of an external MCU or MPU.

#### Note

List of devices that support this command - ATECC608A/B. Refer to device datasheet for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.89 calib\_selftest.c File Reference

CryptoAuthLib Basic API methods for SelfTest command.

```
#include "cryptoauthlib.h"
```

### Functions

- [ATCA\\_STATUS calib\\_selftest](#) ([ATCADevice](#) device, uint8\_t mode, uint16\_t param2, uint8\_t \*result)  
*Executes the SelfTest command, which performs a test of one or more of the cryptographic engines within the ATCA↔ECC608 chip.*

### 20.89.1 Detailed Description

CryptoAuthLib Basic API methods for SelfTest command.

The SelfTest command performs a test of one or more of the cryptographic engines within the device.

#### Note

List of devices that support this command - ATECC608A/B. Refer to device datasheet for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.90 calib\_sha.c File Reference

CryptoAuthLib Basic API methods for SHA command.

```
#include "cryptoauthlib.h"
```

## Data Structures

- struct [hw\\_sha256\\_ctx](#)

## Functions

- [ATCA\\_STATUS calib\\_sha\\_base](#) ([ATCADevice](#) device, [uint8\\_t](#) mode, [uint16\\_t](#) length, [const uint8\\_t](#) \*message, [uint8\\_t](#) \*data\_out, [uint16\\_t](#) \*data\_out\_size)  
*Executes SHA command, which computes a SHA-256 or HMAC/SHA-256 digest for general purpose use by the host system.*
- [ATCA\\_STATUS calib\\_sha\\_start](#) ([ATCADevice](#) device)  
*Executes SHA command to initialize SHA-256 calculation engine.*
- [ATCA\\_STATUS calib\\_sha\\_update](#) ([ATCADevice](#) device, [const uint8\\_t](#) \*message)  
*Executes SHA command to add 64 bytes of message data to the current context.*
- [ATCA\\_STATUS calib\\_sha\\_end](#) ([ATCADevice](#) device, [uint8\\_t](#) \*digest, [uint16\\_t](#) length, [const uint8\\_t](#) \*message)  
*Executes SHA command to complete SHA-256 or HMAC/SHA-256 operation.*
- [ATCA\\_STATUS calib\\_sha\\_read\\_context](#) ([ATCADevice](#) device, [uint8\\_t](#) \*context, [uint16\\_t](#) \*context\_size)  
*Executes SHA command to read the SHA-256 context back. Only for ATECC608 with SHA-256 contexts. HMAC not supported.*
- [ATCA\\_STATUS calib\\_sha\\_write\\_context](#) ([ATCADevice](#) device, [const uint8\\_t](#) \*context, [uint16\\_t](#) context\_size)  
*Executes SHA command to write (restore) a SHA-256 context into the the device. Only supported for ATECC608 with SHA-256 contexts.*
- [ATCA\\_STATUS calib\\_sha](#) ([ATCADevice](#) device, [uint16\\_t](#) length, [const uint8\\_t](#) \*message, [uint8\\_t](#) \*digest)  
*Use the SHA command to compute a SHA-256 digest.*
- [ATCA\\_STATUS calib\\_hw\\_sha2\\_256\\_init](#) ([ATCADevice](#) device, [atca\\_sha256\\_ctx\\_t](#) \*ctx)  
*Initialize a SHA context for performing a hardware SHA-256 operation on a device. Note that only one SHA operation can be run at a time.*
- [ATCA\\_STATUS calib\\_hw\\_sha2\\_256\\_update](#) ([ATCADevice](#) device, [atca\\_sha256\\_ctx\\_t](#) \*ctx, [const uint8\\_t](#) \*data, [size\\_t](#) data\_size)  
*Add message data to a SHA context for performing a hardware SHA-256 operation on a device.*
- [ATCA\\_STATUS calib\\_hw\\_sha2\\_256\\_finish](#) ([ATCADevice](#) device, [atca\\_sha256\\_ctx\\_t](#) \*ctx, [uint8\\_t](#) \*digest)  
*Finish SHA-256 digest for a SHA context for performing a hardware SHA-256 operation on a device.*
- [ATCA\\_STATUS calib\\_hw\\_sha2\\_256](#) ([ATCADevice](#) device, [const uint8\\_t](#) \*data, [size\\_t](#) data\_size, [uint8\\_t](#) \*digest)  
*Use the SHA command to compute a SHA-256 digest.*
- [ATCA\\_STATUS calib\\_sha\\_hmac\\_init](#) ([ATCADevice](#) device, [atca\\_hmac\\_sha256\\_ctx\\_t](#) \*ctx, [uint16\\_t](#) key\_slot)  
*Executes SHA command to start an HMAC/SHA-256 operation.*
- [ATCA\\_STATUS calib\\_sha\\_hmac\\_update](#) ([ATCADevice](#) device, [atca\\_hmac\\_sha256\\_ctx\\_t](#) \*ctx, [const uint8\\_t](#) \*data, [size\\_t](#) data\_size)  
*Executes SHA command to add an arbitrary amount of message data to a HMAC/SHA-256 operation.*
- [ATCA\\_STATUS calib\\_sha\\_hmac\\_finish](#) ([ATCADevice](#) device, [atca\\_hmac\\_sha256\\_ctx\\_t](#) \*ctx, [uint8\\_t](#) \*digest, [uint8\\_t](#) target)  
*Executes SHA command to complete a HMAC/SHA-256 operation.*
- [ATCA\\_STATUS calib\\_sha\\_hmac](#) ([ATCADevice](#) device, [const uint8\\_t](#) \*data, [size\\_t](#) data\_size, [uint16\\_t](#) key\_slot, [uint8\\_t](#) \*digest, [uint8\\_t](#) target)  
*Use the SHA command to compute an HMAC/SHA-256 operation.*

## 20.90.1 Detailed Description

CryptoAuthLib Basic API methods for SHA command.

The SHA command Computes a SHA-256 or HMAC/SHA digest for general purpose use by the host system.

### Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.91 calib\_sign.c File Reference

CryptoAuthLib Basic API methods for Sign command.

```
#include "cryptoauthlib.h"
```

### Functions

- [ATCA\\_STATUS calib\\_sign\\_base](#) (ATCADevice device, uint8\_t mode, uint16\_t key\_id, uint8\_t \*signature)  
*Executes the Sign command, which generates a signature using the ECDSA algorithm.*
- [ATCA\\_STATUS calib\\_sign](#) (ATCADevice device, uint16\_t key\_id, const uint8\_t \*msg, uint8\_t \*signature)  
*Executes Sign command, to sign a 32-byte external message using the private key in the specified slot. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*
- [ATCA\\_STATUS calib\\_sign\\_internal](#) (ATCADevice device, uint16\_t key\_id, bool is\_invalidate, bool is\_full\_sn, uint8\_t \*signature)  
*Executes Sign command to sign an internally generated message.*

## 20.91.1 Detailed Description

CryptoAuthLib Basic API methods for Sign command.

The Sign command generates a signature using the private key in slot with ECDSA algorithm.

### Note

List of devices that support this command - ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.92 calib\_updateextra.c File Reference

CryptoAuthLib Basic API methods for UpdateExtra command.

```
#include "cryptoauthlib.h"
```

### Functions

- [ATCA\\_STATUS calib\\_updateextra](#) (ATCADevice device, uint8\_t mode, uint16\_t new\_value)

*Executes UpdateExtra command to update the values of the two extra bytes within the Configuration zone (bytes 84 and 85).*

### 20.92.1 Detailed Description

CryptoAuthLib Basic API methods for UpdateExtra command.

The UpdateExtra command is used to update the values of the two extra bytes within the Configuration zone after the Configuration zone has been locked.

#### Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.93 calib\_verify.c File Reference

CryptoAuthLib Basic API methods for Verify command.

```
#include "cryptoauthlib.h"  
#include "host/atca_host.h"
```

## Functions

- **ATCA\_STATUS\_calib\_verify** (ATCADevice device, uint8\_t mode, uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*public\_key, const uint8\_t \*other\_data, uint8\_t \*mac)  
*Executes the Verify command, which takes an ECDSA [R,S] signature and verifies that it is correctly generated from a given message and public key. In all cases, the signature is an input to the command.*
- **ATCA\_STATUS\_calib\_verify\_extern** (ATCADevice device, const uint8\_t \*message, const uint8\_t \*signature, const uint8\_t \*public\_key, bool \*is\_verified)  
*Executes the Verify command, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*
- **ATCA\_STATUS\_calib\_verify\_extern\_mac** (ATCADevice device, const uint8\_t \*message, const uint8\_t \*signature, const uint8\_t \*public\_key, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)  
*Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with all components (message, signature, and public key) supplied. This function is only available on the ATECC608.*
- **ATCA\_STATUS\_calib\_verify\_stored** (ATCADevice device, const uint8\_t \*message, const uint8\_t \*signature, uint16\_t key\_id, bool \*is\_verified)  
*Executes the Verify command, which verifies a signature (ECDSA verify operation) with a public key stored in the device. The message to be signed will be loaded into the Message Digest Buffer to the ATECC608 device or TempKey for other devices.*
- **ATCA\_STATUS\_calib\_verify\_stored\_mac** (ATCADevice device, const uint8\_t \*message, const uint8\_t \*signature, uint16\_t key\_id, const uint8\_t \*num\_in, const uint8\_t \*io\_key, bool \*is\_verified)  
*Executes the Verify command with verification MAC, which verifies a signature (ECDSA verify operation) with a public key stored in the device. This function is only available on the ATECC608.*
- **ATCA\_STATUS\_calib\_verify\_validate** (ATCADevice device, uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*other\_data, bool \*is\_verified)  
*Executes the Verify command in Validate mode to validate a public key stored in a slot.*
- **ATCA\_STATUS\_calib\_verify\_invalidate** (ATCADevice device, uint16\_t key\_id, const uint8\_t \*signature, const uint8\_t \*other\_data, bool \*is\_verified)  
*Executes the Verify command in Invalidate mode which invalidates a previously validated public key stored in a slot.*

### 20.93.1 Detailed Description

CryptoAuthLib Basic API methods for Verify command.

The Verify command takes an ECDSA [R,S] signature and verifies that it is correctly generated given an input message digest and public key.

#### Note

List of devices that support this command - ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheet for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.94 calib\_write.c File Reference

CryptoAuthLib Basic API methods for Write command.

```
#include "cryptoauthlib.h"
#include "host/atca_host.h"
```

## Functions

- **ATCA\_STATUS calib\_write** (ATCADevice device, uint8\_t zone, uint16\_t address, const uint8\_t \*value, const uint8\_t \*mac)  
*Executes the Write command, which writes either one four byte word or a 32-byte block to one of the EEPROM zones on the device. Depending upon the value of the WriteConfig byte for this slot, the data may be required to be encrypted by the system prior to being sent to the device. This command cannot be used to write slots configured as ECC private keys.*
- **ATCA\_STATUS calib\_write\_zone** (ATCADevice device, uint8\_t zone, uint16\_t slot, uint8\_t block, uint8\_t offset, const uint8\_t \*data, uint8\_t len)  
*Executes the Write command, which writes either 4 or 32 bytes of data into a device zone.*
- **ATCA\_STATUS calib\_write\_enc** (ATCADevice device, uint16\_t key\_id, uint8\_t block, const uint8\_t \*data, const uint8\_t \*enc\_key, const uint16\_t enc\_key\_id, const uint8\_t num\_in[NONCE\_NUMIN\_SIZE])  
*Executes the Write command, which performs an encrypted write of a 32 byte block into given slot.*
- **ATCA\_STATUS calib\_write\_config\_zone** (ATCADevice device, const uint8\_t \*config\_data)  
*Executes the Write command, which writes the configuration zone.*
- **ATCA\_STATUS calib\_write\_pubkey** (ATCADevice device, uint16\_t slot, const uint8\_t \*public\_key)  
*Uses the write command to write a public key to a slot in the proper format.*
- **ATCA\_STATUS calib\_write\_bytes\_zone** (ATCADevice device, uint8\_t zone, uint16\_t slot, size\_t offset\_bytes, const uint8\_t \*data, size\_t length)  
*Executes the Write command, which writes data into the configuration, otp, or data zones with a given byte offset and length. Offset and length must be multiples of a word (4 bytes).*
- **ATCA\_STATUS calib\_write\_config\_counter** (ATCADevice device, uint16\_t counter\_id, uint32\_t counter\_value)  
*Initialize one of the monotonic counters in device with a specific value.*

### 20.94.1 Detailed Description

CryptoAuthLib Basic API methods for Write command.

The Write command writes either one 4-byte word or a 32-byte block to one of the EEPROM zones on the device. Depending upon the value of the WriteConfig byte for a slot, the data may be required to be encrypted by the system prior to being sent to the device

#### Note

List of devices that support this command - ATSHA204A, ATECC108A, ATECC508A, and ATECC608A/B. There are differences in the modes that they support. Refer to device datasheets for full details.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.94.2 Function Documentation



### 20.94.2.1 calib\_write\_enc()

```
ATCA_STATUS calib_write_enc (
    ATCADevice device,
    uint16_t key_id,
    uint8_t block,
    const uint8_t * data,
    const uint8_t * enc_key,
    const uint16_t enc_key_id,
    const uint8_t num_in[NONCE_NUMIN_SIZE] )
```

Executes the Write command, which performs an encrypted write of a 32 byte block into given slot.

The function takes clear text bytes and encrypts them for writing over the wire. Data zone must be locked and the slot configuration must be set to encrypted write for the block to be successfully written.

#### Parameters

in	<i>device</i>	Device context pointer
in	<i>key_id</i>	Slot ID to write to.
in	<i>block</i>	Index of the 32 byte block to write in the slot.
in	<i>data</i>	32 bytes of clear text data to be written to the slot
in	<i>enc_key</i>	WriteKey to encrypt with for writing
in	<i>enc_key_id</i>	The KeyID of the WriteKey
in	<i>num_in</i>	20 byte host nonce to inject into Nonce calculation

returns ATCA\_SUCCESS on success, otherwise an error code.

## 20.95 cryptoauthlib.h File Reference

Single aggregation point for all CryptoAuthLib header files.

```
#include <stdio.h>
#include <stdint.h>
#include <stddef.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include "atca_config.h"
#include "atca_compiler.h"
#include "atca_version.h"
#include "atca_status.h"
#include "atca_debug.h"
#include "atca_iface.h"
#include "atca_helpers.h"
#include "hal/atca_hal.h"
#include "atca_cfgs.h"
#include "atca_device.h"
#include "calib/calib_basic.h"
#include "calib/calib_command.h"
#include "calib/calib_aes_gcm.h"
```

```
#include "talib/talib_status.h"
#include "talib/talib_basic.h"
#include "atca_basic.h"
```

### Macros

- #define [ATCA\\_SHA\\_SUPPORT](#) 1
- #define [ATCA\\_ATECC608\\_SUPPORT](#)
- #define [ATCA\\_ECC\\_SUPPORT](#) 1
- #define [ATCA\\_CA\\_SUPPORT](#) 1
- #define [ATCA\\_TA\\_SUPPORT](#) 1
- #define [ATCA\\_DLL](#) SHARED\_LIB\_IMPORT
- #define [ATCA\\_SHA256\\_BLOCK\\_SIZE](#) (64)
- #define [ATCA\\_SHA256\\_DIGEST\\_SIZE](#) (32)
- #define [ATCA\\_AES128\\_BLOCK\\_SIZE](#) (16)
- #define [ATCA\\_AES128\\_KEY\\_SIZE](#) (16)
- #define [ATCA\\_ECCP256\\_KEY\\_SIZE](#) (32)
- #define [ATCA\\_ECCP256\\_PUBKEY\\_SIZE](#) (64)
- #define [ATCA\\_ECCP256\\_SIG\\_SIZE](#) (64)
- #define [ATCA\\_ZONE\\_CONFIG](#) ((uint8\_t)0x00)
- #define [ATCA\\_ZONE\\_OTP](#) ((uint8\_t)0x01)
- #define [ATCA\\_ZONE\\_DATA](#) ((uint8\_t)0x02)
- #define [SHA\\_MODE\\_TARGET\\_TEMPKEY](#) ((uint8\_t)0x00)
- #define [SHA\\_MODE\\_TARGET\\_MSGDIGBUF](#) ((uint8\_t)0x40)
- #define [SHA\\_MODE\\_TARGET\\_OUT\\_ONLY](#) ((uint8\_t)0xC0)
- #define [ATCA\\_STRINGIFY](#)(x) #x
- #define [ATCA\\_TOSTRING](#)(x) [ATCA\\_STRINGIFY](#)(x)
- #define [ATCA\\_TRACE](#)(s, m) [atca\\_trace](#)(s)

### 20.95.1 Detailed Description

Single aggregation point for all CryptoAuthLib header files.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.95.2 Macro Definition Documentation

#### 20.95.2.1 ATCA\_AES128\_BLOCK\_SIZE

```
#define ATCA_AES128_BLOCK_SIZE (16)
```

**20.95.2.2 ATCA\_AES128\_KEY\_SIZE**

```
#define ATCA_AES128_KEY_SIZE (16)
```

**20.95.2.3 ATCA\_ATECC608\_SUPPORT**

```
#define ATCA_ATECC608_SUPPORT
```

**20.95.2.4 ATCA\_CA\_SUPPORT**

```
#define ATCA_CA_SUPPORT 1
```

**20.95.2.5 ATCA\_DLL**

```
#define ATCA_DLL SHARED_LIB_IMPORT
```

**20.95.2.6 ATCA\_ECC\_SUPPORT**

```
#define ATCA_ECC_SUPPORT 1
```

**20.95.2.7 ATCA\_ECCP256\_KEY\_SIZE**

```
#define ATCA_ECCP256_KEY_SIZE (32)
```

**20.95.2.8 ATCA\_ECCP256\_PUBKEY\_SIZE**

```
#define ATCA_ECCP256_PUBKEY_SIZE (64)
```

**20.95.2.9 ATCA\_ECCP256\_SIG\_SIZE**

```
#define ATCA_ECCP256_SIG_SIZE (64)
```

### 20.95.2.10 ATCA\_SHA256\_BLOCK\_SIZE

```
#define ATCA_SHA256_BLOCK_SIZE (64)
```

### 20.95.2.11 ATCA\_SHA256\_DIGEST\_SIZE

```
#define ATCA_SHA256_DIGEST_SIZE (32)
```

### 20.95.2.12 ATCA\_SHA\_SUPPORT

```
#define ATCA_SHA_SUPPORT 1
```

Library Configuration File - All build attributes should be included in atca\_config.h

### 20.95.2.13 ATCA\_STRINGIFY

```
#define ATCA_STRINGIFY(  
    x ) #x
```

### 20.95.2.14 ATCA\_TA\_SUPPORT

```
#define ATCA_TA_SUPPORT 1
```

### 20.95.2.15 ATCA\_TOSTRING

```
#define ATCA_TOSTRING(  
    x ) ATCA_STRINGIFY(x)
```

### 20.95.2.16 ATCA\_TRACE

```
#define ATCA_TRACE(  
    s,  
    m ) atca_trace(s)
```

#### 20.95.2.17 ATCA\_ZONE\_CONFIG

```
#define ATCA_ZONE_CONFIG ((uint8_t)0x00)
```

#### 20.95.2.18 ATCA\_ZONE\_DATA

```
#define ATCA_ZONE_DATA ((uint8_t)0x02)
```

#### 20.95.2.19 ATCA\_ZONE\_OTP

```
#define ATCA_ZONE_OTP ((uint8_t)0x01)
```

#### 20.95.2.20 SHA\_MODE\_TARGET\_MSGDIGBUF

```
#define SHA_MODE_TARGET_MSGDIGBUF ((uint8_t)0x40)
```

Place resulting digest both in Output buffer and Message Digest Buffer

#### 20.95.2.21 SHA\_MODE\_TARGET\_OUT\_ONLY

```
#define SHA_MODE_TARGET_OUT_ONLY ((uint8_t)0xC0)
```

Place resulting digest both in Output buffer ONLY

#### 20.95.2.22 SHA\_MODE\_TARGET\_TEMPKEY

```
#define SHA_MODE_TARGET_TEMPKEY ((uint8_t)0x00)
```

Place resulting digest both in Output buffer and TempKey

## 20.96 cryptoki.h File Reference

```
#include "pkcs11.h"
```

## Macros

- `#define PKCS11_HELPER_DLL_IMPORT`
- `#define PKCS11_HELPER_DLL_EXPORT`
- `#define PKCS11_HELPER_DLL_LOCAL`
- `#define PKCS11_API`
- `#define PKCS11_LOCAL PKCS11_HELPER_DLL_LOCAL`
- `#define CK_PTR *`
- `#define CK_DECLARE_FUNCTION(returnType, name) returnType PKCS11_API name`
- `#define CK_DECLARE_FUNCTION_POINTER(returnType, name) returnType PKCS11_API(*name)`
- `#define CK_CALLBACK_FUNCTION(returnType, name) returnType(*name)`
- `#define NULL_PTR 0`

### 20.96.1 Macro Definition Documentation

#### 20.96.1.1 CK\_CALLBACK\_FUNCTION

```
#define CK_CALLBACK_FUNCTION(  
    returnType,  
    name ) returnType(*name)
```

#### 20.96.1.2 CK\_DECLARE\_FUNCTION

```
#define CK_DECLARE_FUNCTION(  
    returnType,  
    name ) returnType PKCS11_API name
```

#### 20.96.1.3 CK\_DECLARE\_FUNCTION\_POINTER

```
#define CK_DECLARE_FUNCTION_POINTER(  
    returnType,  
    name ) returnType PKCS11_API(*name)
```

#### 20.96.1.4 CK\_PTR

```
#define CK_PTR *
```

### 20.96.1.5 NULL\_PTR

```
#define NULL_PTR 0
```

### 20.96.1.6 PKCS11\_API

```
#define PKCS11_API
```

### 20.96.1.7 PKCS11\_HELPER\_DLL\_EXPORT

```
#define PKCS11_HELPER_DLL_EXPORT
```

### 20.96.1.8 PKCS11\_HELPER\_DLL\_IMPORT

```
#define PKCS11_HELPER_DLL_IMPORT
```

### 20.96.1.9 PKCS11\_HELPER\_DLL\_LOCAL

```
#define PKCS11_HELPER_DLL_LOCAL
```

### 20.96.1.10 PKCS11\_LOCAL

```
#define PKCS11_LOCAL PKCS11_HELPER_DLL_LOCAL
```

## 20.97 example\_cert\_chain.c File Reference

```
#include "atcacert/atcacert_def.h"  
#include "example_cert_chain.h"
```

### Variables

- const [atcacert\\_def\\_t g\\_cert\\_def\\_0\\_root](#)
- const [atcacert\\_cert\\_element\\_t g\\_cert\\_elements\\_1\\_signer \[\]](#)
- const [uint8\\_t g\\_cert\\_template\\_1\\_signer \[\]](#)
- const [atcacert\\_def\\_t g\\_cert\\_def\\_1\\_signer](#)
- const [uint8\\_t g\\_cert\\_template\\_2\\_device \[\]](#)
- const [atcacert\\_def\\_t g\\_cert\\_def\\_2\\_device](#)

## 20.97.1 Variable Documentation

### 20.97.1.1 g\_cert\_def\_0\_root

```
const atcacert_def_t g_cert_def_0_root
```

#### Initial value:

```
= {  
    .type           = CERTTYPE_X509,  
    .template_id    = 0,  
    .public_key_dev_loc = {  
        .zone       = DEVZONE_DATA,  
        .slot       = 15,  
        .is_genkey   = 0,  
        .offset      = 0,  
        .count       = 72  
    }  
}
```

### 20.97.1.2 g\_cert\_def\_1\_signer

```
const atcacert_def_t g_cert_def_1_signer
```

### 20.97.1.3 g\_cert\_def\_2\_device

```
const atcacert_def_t g_cert_def_2_device
```

### 20.97.1.4 g\_cert\_elements\_1\_signer

```
const atcacert_cert_element_t g_cert_elements_1_signer[]
```

### 20.97.1.5 g\_cert\_template\_1\_signer

```
const uint8_t g_cert_template_1_signer[]
```



### 20.97.1.6 g\_cert\_template\_2\_device

```
const uint8_t g_cert_template_2_device[]
```

Initial value:

```
= {
    0x30, 0x82, 0x01, 0xa6, 0x30, 0x82, 0x01, 0x4b, 0xa0, 0x03, 0x02, 0x01, 0x02, 0x02, 0x10, 0x41,
    0xa6, 0x8b, 0xe4, 0x36, 0xdd, 0xc3, 0xd8, 0x39, 0xfa, 0xbd, 0xd7, 0x27, 0xd9, 0x74, 0xe7, 0x30,
    0x0a, 0x06, 0x08, 0x2a, 0x86, 0x48, 0xce, 0x3d, 0x04, 0x03, 0x02, 0x30, 0x34, 0x31, 0x14, 0x30,
    0x12, 0x06, 0x03, 0x55, 0x04, 0x0a, 0x0c, 0x0b, 0x45, 0x78, 0x61, 0x6d, 0x70, 0x6c, 0x65, 0x20,
    0x49, 0x6e, 0x63, 0x31, 0x1c, 0x30, 0x1a, 0x06, 0x03, 0x55, 0x04, 0x03, 0x0c, 0x13, 0x45, 0x78,
    0x61, 0x6d, 0x70, 0x6c, 0x65, 0x20, 0x53, 0x69, 0x67, 0x6e, 0x65, 0x72, 0x20, 0x46, 0x46, 0x46,
    0x46, 0x30, 0x20, 0x17, 0x0d, 0x31, 0x37, 0x30, 0x37, 0x31, 0x30, 0x32, 0x30, 0x30, 0x30, 0x30,
    0x30, 0x5a, 0x18, 0x0f, 0x33, 0x30, 0x30, 0x30, 0x31, 0x32, 0x33, 0x31, 0x32, 0x33, 0x35, 0x39,
    0x35, 0x39, 0x5a, 0x30, 0x2f, 0x31, 0x14, 0x30, 0x12, 0x06, 0x03, 0x55, 0x04, 0x0a, 0x0c, 0x0b,
    0x45, 0x78, 0x61, 0x6d, 0x70, 0x6c, 0x65, 0x20, 0x49, 0x6e, 0x63, 0x31, 0x17, 0x30, 0x15, 0x06,
    0x03, 0x55, 0x04, 0x03, 0x0c, 0x0e, 0x45, 0x78, 0x61, 0x6d, 0x70, 0x6c, 0x65, 0x20, 0x44, 0x65,
    0x76, 0x69, 0x63, 0x65, 0x30, 0x59, 0x30, 0x13, 0x06, 0x07, 0x2a, 0x86, 0x48, 0xce, 0x3d, 0x02,
    0x01, 0x06, 0x08, 0x2a, 0x86, 0x48, 0xce, 0x3d, 0x03, 0x01, 0x07, 0x03, 0x42, 0x00, 0x04, 0x96,
    0x27, 0xf1, 0x3e, 0x80, 0xac, 0xf9, 0xd4, 0x12, 0xce, 0x3b, 0x0d, 0x68, 0xf7, 0x4e, 0xb2, 0xc6,
    0x07, 0x35, 0x00, 0xb7, 0x78, 0x5b, 0xac, 0xe6, 0x50, 0x30, 0x54, 0x77, 0x7f, 0xc8, 0x62, 0x21,
    0xce, 0xf2, 0x5a, 0x9a, 0x9e, 0x86, 0x40, 0xc2, 0x29, 0xd6, 0x4a, 0x32, 0x1e, 0xb9, 0x4a, 0x1b,
    0x1c, 0x94, 0xf5, 0x39, 0x88, 0xae, 0xfe, 0x49, 0xcc, 0xfd, 0xbf, 0x8a, 0x0d, 0x34, 0xb8, 0xa3,
    0x42, 0x30, 0x40, 0x30, 0x1d, 0x06, 0x03, 0x55, 0x1d, 0x0e, 0x04, 0x16, 0x04, 0x14, 0x2d, 0xda,
    0x6c, 0x36, 0xd5, 0xa5, 0x5a, 0xce, 0x97, 0x10, 0x3d, 0xbb, 0xaf, 0x9c, 0x66, 0x2a, 0xcd, 0x3e,
    0xe6, 0xcf, 0x30, 0x1f, 0x06, 0x03, 0x55, 0x1d, 0x23, 0x04, 0x18, 0x30, 0x16, 0x80, 0x14, 0xc6,
    0x70, 0xe0, 0x5e, 0x8a, 0x45, 0x0d, 0xb8, 0x2c, 0x00, 0x2a, 0x40, 0x06, 0x39, 0x4c, 0x19, 0x58,
    0x04, 0x35, 0x76, 0x30, 0x0a, 0x06, 0x08, 0x2a, 0x86, 0x48, 0xce, 0x3d, 0x04, 0x03, 0x02, 0x03,
    0x49, 0x00, 0x30, 0x46, 0x02, 0x21, 0x00, 0xe1, 0xfc, 0x00, 0x23, 0xc1, 0x3d, 0x01, 0x3f, 0x22,
    0x31, 0x0b, 0xf0, 0xb8, 0xf4, 0xf4, 0x22, 0xfc, 0x95, 0x96, 0x33, 0x9c, 0xb9, 0x62, 0xb1, 0xfc,
    0x8a, 0x2d, 0xa8, 0x5c, 0xee, 0x67, 0x72, 0x02, 0x21, 0x00, 0xa1, 0x0d, 0x47, 0xe4, 0xfd, 0x0d,
    0x15, 0xd8, 0xde, 0xa1, 0xb5, 0x96, 0x28, 0x4e, 0x7a, 0x0b, 0xbe, 0xcc, 0xec, 0xe8, 0x8e, 0xcc,
    0x7a, 0x31, 0xb3, 0x00, 0x8b, 0xc0, 0x2e, 0x4f, 0x99, 0xc5
}
```

## 20.98 example\_cert\_chain.h File Reference

```
#include "atcacert/atcacert_def.h"
```

### Variables

- const [atcacert\\_def\\_t g\\_cert\\_def\\_1\\_signer](#)
- const [atcacert\\_def\\_t g\\_cert\\_def\\_2\\_device](#)

### 20.98.1 Variable Documentation

#### 20.98.1.1 g\_cert\_def\_1\_signer

```
const atcacert\_def\_t g_cert_def_1_signer
```

#### 20.98.1.2 g\_cert\_def\_2\_device

```
const atcacert\_def\_t g_cert_def_2_device
```

## 20.99 example\_pkcs11\_config.c File Reference

```
#include "cryptoauthlib.h"
#include "pkcs11_config.h"
#include "pkcs11/pkcs11_object.h"
#include "pkcs11/pkcs11_slot.h"
#include "example_cert_chain.h"
```

### Macros

- `#define pkcs11configLABEL_DEVICE_CERTIFICATE_FOR_TLS "device"`
- `#define pkcs11configLABEL_JITP_CERTIFICATE "signer"`
- `#define pkcs11configLABEL_DEVICE_PRIVATE_KEY_FOR_TLS pkcs11configLABEL_DEVICE_CERTIFICATE_FOR_TLS`
- `#define pkcs11configLABEL_DEVICE_PUBLIC_KEY_FOR_TLS pkcs11configLABEL_DEVICE_CERTIFICATE_FOR_TLS`

### Functions

- `CK_RV pkcs11_config_cert (pkcs11_lib_ctx_ptr pLibCtx, pkcs11_slot_ctx_ptr pSlot, pkcs11_object_ptr pObject, CK_ATTRIBUTE_PTR pLabel)`
- `CK_RV pkcs11_config_key (pkcs11_lib_ctx_ptr pLibCtx, pkcs11_slot_ctx_ptr pSlot, pkcs11_object_ptr pObject, CK_ATTRIBUTE_PTR pLabel)`
- `CK_RV pkcs11_config_load_objects (pkcs11_slot_ctx_ptr pSlot)`

### Variables

- `const uint8_t atecc608_config []`

## 20.99.1 Macro Definition Documentation

### 20.99.1.1 pkcs11configLABEL\_DEVICE\_CERTIFICATE\_FOR\_TLS

```
#define pkcs11configLABEL_DEVICE_CERTIFICATE_FOR_TLS "device"
```

### 20.99.1.2 pkcs11configLABEL\_DEVICE\_PRIVATE\_KEY\_FOR\_TLS

```
#define pkcs11configLABEL_DEVICE_PRIVATE_KEY_FOR_TLS pkcs11configLABEL_DEVICE_CERTIFICATE_FOR_TLS
```

### 20.99.1.3 pkcs11configLABEL\_DEVICE\_PUBLIC\_KEY\_FOR\_TLS

```
#define pkcs11configLABEL_DEVICE_PUBLIC_KEY_FOR_TLS pkcs11configLABEL_DEVICE_CERTIFICATE_FOR_TLS
```

### 20.99.1.4 pkcs11configLABEL\_JITP\_CERTIFICATE

```
#define pkcs11configLABEL_JITP_CERTIFICATE "signer"
```

## 20.99.2 Function Documentation

### 20.99.2.1 pkcs11\_config\_cert()

```
CK_RV pkcs11_config_cert (
    pkcs11_lib_ctx_ptr pLibCtx,
    pkcs11_slot_ctx_ptr pSlot,
    pkcs11_object_ptr pObject,
    CK_ATTRIBUTE_PTR pLabel )
```

### 20.99.2.2 pkcs11\_config\_key()

```
CK_RV pkcs11_config_key (
    pkcs11_lib_ctx_ptr pLibCtx,
    pkcs11_slot_ctx_ptr pSlot,
    pkcs11_object_ptr pObject,
    CK_ATTRIBUTE_PTR pLabel )
```

### 20.99.2.3 pkcs11\_config\_load\_objects()

```
CK_RV pkcs11_config_load_objects (
    pkcs11_slot_ctx_ptr pSlot )
```

## 20.99.3 Variable Documentation

### 20.99.3.1 atecc608\_config

```
const uint8_t atecc608_config[]
```

#### Initial value:

```
= {  
    0x01, 0x23, 0x00, 0x00, 0x00, 0x00, 0x60, 0x01, 0x00, 0x00, 0x00, 0x00, 0xEE, 0x01, 0x01, 0x00,  
    0xC0, 0x00, 0x00, 0x01, 0x8F, 0x20, 0xC4, 0x44, 0x87, 0x20, 0x87, 0x20, 0x8F, 0x0F, 0xC4, 0x36,  
    0x9F, 0x0F, 0x82, 0x20, 0x0F, 0x0F, 0xC4, 0x44, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F,  
    0x0F, 0x0F, 0x0F, 0x0F, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF,  
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
    0x33, 0x00, 0x1C, 0x00, 0x13, 0x00, 0x13, 0x00, 0x7C, 0x00, 0x1C, 0x00, 0x3C, 0x00, 0x33, 0x00,  
    0x3C, 0x00, 0x3C, 0x00, 0x3C, 0x00, 0x30, 0x00, 0x3C, 0x00, 0x3C, 0x00, 0x3C, 0x00, 0x30, 0x00,  
}
```

Standard Configuration Structure for ATECC608 devices

## 20.100 hal\_all\_platforms\_kit\_hidapi.c File Reference

HAL for kit protocol over HID for any platform.

```
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include "hidapi.h"  
#include "atca_hal.h"  
#include "hal_all_platforms_kit_hidapi.h"  
#include "hal/kit_protocol.h"
```

### Functions

- [ATCA\\_STATUS hal\\_kit\\_hid\\_discover\\_buses](#) (int hid\_buses[], int max\_buses)  
*discover cdc buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_discover\\_devices](#) (int bus\_num, [ATCAIfaceCfg](#) cfg[], int \*found)  
*discover any CryptoAuth devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_init](#) (void \*hal, [ATCAIfaceCfg](#) \*cfg)  
*HAL implementation of Kit USB HID init.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_post\\_init](#) ([ATCAIface](#) iface)  
*HAL implementation of Kit HID post init.*
- [ATCA\\_STATUS kit\\_phy\\_send](#) ([ATCAIface](#) iface, uint8\_t \*txdata, int txlength)  
*HAL implementation of send over USB HID.*
- [ATCA\\_STATUS kit\\_phy\\_receive](#) ([ATCAIface](#) iface, uint8\_t \*rxdata, int \*rxsize)  
*HAL implementation of kit protocol send over USB HID.*
- [ATCA\\_STATUS kit\\_phy\\_num\\_found](#) (int8\_t \*num\_found)  
*Number of USB HID devices found.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_send](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of kit protocol send over USB HID.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_receive](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxsize)  
*HAL implementation of send over USB HID.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_wake](#) ([ATCAIface](#) iface)

*Call the wake for kit protocol.*

- [ATCA\\_STATUS hal\\_kit\\_hid\\_idle](#) ([ATCAIface](#) iface)

*Call the idle for kit protocol.*

- [ATCA\\_STATUS hal\\_kit\\_hid\\_sleep](#) ([ATCAIface](#) iface)

*Call the sleep for kit protocol.*

- [ATCA\\_STATUS hal\\_kit\\_hid\\_release](#) (void \*hal\_data)

*Close the physical port for HID.*

## Variables

- [atcahid\\_t \\_gHid](#)

### 20.100.1 Detailed Description

HAL for kit protocol over HID for any platform.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.101 hal\_all\_platforms\_kit\_hidapi.h File Reference

HAL for kit protocol over HID for any platform.

```
#include "hidapi.h"
```

## Data Structures

- struct [atcahid](#)

## Macros

- #define [HID\\_DEVICES\\_MAX](#) 10
- #define [HID\\_PACKET\\_MAX](#) 512

## Typedefs

- typedef struct [atcahid](#) [atcahid\\_t](#)

### 20.101.1 Detailed Description

HAL for kit protocol over HID for any platform.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.102 hal\_esp32\_i2c.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <driver/i2c.h>
#include "esp_err.h"
#include "esp_log.h"
#include "cryptoauthlib.h"
```

### Data Structures

- struct [atcal2Cmaster](#)  
*this is the hal\_data for ATCA HAL for ASF SERCOM*

### Macros

- #define [SDA\\_PIN](#) 16
- #define [SCL\\_PIN](#) 17
- #define [ACK\\_CHECK\\_EN](#) 0x1
- #define [ACK\\_CHECK\\_DIS](#) 0x0
- #define [ACK\\_VAL](#) 0x0
- #define [NACK\\_VAL](#) 0x1
- #define [LOG\\_LOCAL\\_LEVEL](#) ESP\_LOG\_INFO
- #define [MAX\\_I2C\\_BUSES](#) 2

### Typedefs

- typedef struct [atcal2Cmaster](#) [ATCAI2CMaster\\_t](#)

### Functions

- void [hal\\_i2c\\_change\\_baud](#) ([ATCAIface](#) iface, uint32\_t speed)  
*method to change the bus speed of I2C*
- [ATCA\\_STATUS](#) [hal\\_i2c\\_init](#) (void \*hal, [ATCAIfaceCfg](#) \*cfg)  
*hal\_i2c\_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIFace instances using the same bus, and you can have multiple ATCAIFace instances on multiple i2c buses, so hal\_i2c\_init manages these things and ATCAIFace is abstracted from the physical details.*
- [ATCA\\_STATUS](#) [hal\\_i2c\\_post\\_init](#) ([ATCAIface](#) iface)  
*HAL implementation of I2C post init.*
- [ATCA\\_STATUS](#) [hal\\_i2c\\_send](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of I2C send.*
- [ATCA\\_STATUS](#) [hal\\_i2c\\_receive](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)  
*HAL implementation of I2C receive function.*
- [ATCA\\_STATUS](#) [hal\\_i2c\\_release](#) (void \*hal\_data)  
*manages reference count on given bus and releases resource if no more references exist*
- [ATCA\\_STATUS](#) [hal\\_i2c\\_wake](#) ([ATCAIface](#) iface)  
*wake up CryptoAuth device using I2C bus*

- [ATCA\\_STATUS hal\\_i2c\\_idle](#) ([ATCAIface](#) iface)  
*idle CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_sleep](#) ([ATCAIface](#) iface)  
*sleep CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_discover\\_buses](#) (int i2c\_buses[], int max\_buses)  
*discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge*
- [ATCA\\_STATUS hal\\_i2c\\_discover\\_devices](#) (int bus\_num, [ATCAIfaceCfg](#) \*cfg, int \*found)  
*discover any CryptoAuth devices on a given logical bus number*

## Variables

- [ATCAI2CMaster\\_t](#) \* [i2c\\_hal\\_data](#) [2]
- int [i2c\\_bus\\_ref\\_ct](#) = 0
- [i2c\\_config\\_t](#) [conf](#)
- const char \* [TAG](#) = "HAL\_I2C"

## 20.102.1 Macro Definition Documentation

### 20.102.1.1 ACK\_CHECK\_DIS

```
#define ACK_CHECK_DIS 0x0
```

I2C master will not check ack from slave

### 20.102.1.2 ACK\_CHECK\_EN

```
#define ACK_CHECK_EN 0x1
```

I2C master will check ack from slave

### 20.102.1.3 ACK\_VAL

```
#define ACK_VAL 0x0
```

I2C ack value

### 20.102.1.4 LOG\_LOCAL\_LEVEL

```
#define LOG_LOCAL_LEVEL ESP_LOG_INFO
```

### 20.102.1.5 MAX\_I2C\_BUSES

```
#define MAX_I2C_BUSES 2
```

### 20.102.1.6 NACK\_VAL

```
#define NACK_VAL 0x1
```

I2C nack value

### 20.102.1.7 SCL\_PIN

```
#define SCL_PIN 17
```

### 20.102.1.8 SDA\_PIN

```
#define SDA_PIN 16
```

## 20.102.2 Typedef Documentation

### 20.102.2.1 ATCAI2CMaster\_t

```
typedef struct atcaI2Cmaster ATCAI2CMaster_t
```

## 20.102.3 Function Documentation

### 20.102.3.1 hal\_i2c\_change\_baud()

```
void hal_i2c_change_baud (
    ATCAIface iface,
    uint32_t speed )
```

method to change the bus speec of I2C



## Parameters

in	<i>iface</i>	interface on which to change bus speed
in	<i>speed</i>	baud rate (typically 100000 or 400000)

**20.102.3.2 hal\_i2c\_discover\_buses()**

```
ATCA_STATUS hal_i2c_discover_buses (
    int i2c_buses[],
    int max_buses )
```

discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge

This HAL implementation assumes you've included the ASF TWI libraries in your project, otherwise, the HAL layer will not compile because the ASF TWI drivers are a dependency.

logical to physical bus mapping structure

discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge. This function is not implemented.

## Parameters

in	<i>i2c_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover

## Returns

ATCA\_SUCCESS

## Parameters

in	<i>i2c_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover

## Returns

ATCA\_UNIMPLEMENTED

discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge

## Parameters

in	<i>i2c_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover

### Returns

ATCA\_SUCCESS

discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge

### Parameters

in	<i>i2c_buses</i>	- an array of logical bus numbers
in	<i>max_buses</i>	- maximum number of buses the app wants to attempt to discover return ATCA_SUCCESS

### 20.102.3.3 hal\_i2c\_discover\_devices()

```
ATCA_STATUS hal_i2c_discover_devices (
    int bus_num,
    ATCAIfaceCfg * cfg,
    int * found )
```

discover any CryptoAuth devices on a given logical bus number

### Parameters

in	<i>bus_num</i>	logical bus number on which to look for CryptoAuth devices
out	<i>cfg</i>	pointer to head of an array of interface config structures which get filled in by this method
out	<i>found</i>	number of devices found on this bus

### Returns

ATCA\_SUCCESS

### 20.102.3.4 hal\_i2c\_idle()

```
ATCA_STATUS hal_i2c_idle (
    ATCAIface iface )
```

idle CryptoAuth device using I2C bus

### Parameters

in	<i>iface</i>	interface to logical device to idle
----	--------------	-------------------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

< I2C master will check ack from slave

< I2C master will not check ack from slave

### 20.102.3.5 hal\_i2c\_init()

```
ATCA_STATUS hal_i2c_init (
    void * hal,
    ATCAIFaceCfg * cfg )
```

hal\_i2c\_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIFace instances using the same bus, and you can have multiple ATCAIFace instances on multiple i2c buses, so hal\_i2c\_init manages these things and ATCAIFace is abstracted from the physical details.

hal\_i2c\_init manages requests to initialize a physical interface. It manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIFace instances using the same bus, and you can have multiple ATCAIFace instances on multiple i2c buses, so hal\_i2c\_init manages these things and ATCAIFace is abstracted from the physical details.

initialize an I2C interface using given config

HAL implementation of I2C init.

- this HAL implementation assumes you've included the START Twi libraries in your project, otherwise, the HAL layer will not compile because the START TWI drivers are a dependency \*

initialize an I2C interface using given config

#### Parameters

in	hal	- opaque ptr to HAL data
in	cfg	- interface configuration

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

this implementation assumes I2C peripheral has been enabled by user. It only initialize an I2C interface using given config.

#### Parameters

in	hal	pointer to HAL specific data that is maintained by this HAL
in	cfg	pointer to HAL specific configuration data that is used to initialize this HAL

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

- this HAL implementation assumes you've included the ASF SERCOM I2C libraries in your project, otherwise, the HAL layer will not compile because the ASF I2C drivers are a dependency \*

**Parameters**

in	<i>hal</i>	- opaque ptr to HAL data
in	<i>cfg</i>	- interface configuration

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

initialize an I2C interface using given config

**Parameters**

in	<i>hal</i>	- opaque ptr to HAL data
in	<i>cfg</i>	- interface configuration

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

- this HAL implementation assumes you've included the ASF Twi libraries in your project, otherwise, the HAL layer will not compile because the ASF TWI drivers are a dependency \*

initialize an I2C interface using given config

**Parameters**

in	<i>hal</i>	- opaque ptr to HAL data
in	<i>cfg</i>	- interface configuration

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.102.3.6 hal\_i2c\_post\_init()**

```
ATCA_STATUS hal_i2c_post_init (  
    ATCAIface iface )
```

HAL implementation of I2C post init.

**Parameters**

in	<i>iface</i>	instance
----	--------------	----------

**Returns**

ATCA\_SUCCESS

**Parameters**

in	<i>iface</i>	instance
----	--------------	----------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**20.102.3.7 hal\_i2c\_receive()**

```
ATCA_STATUS hal_i2c_receive (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * rxdata,
    uint16_t * rxlength )
```

HAL implementation of I2C receive function.

HAL implementation of I2C receive function for ASF I2C.

HAL implementation of I2C receive function for START I2C.

**Parameters**

in	<i>iface</i>	Device to interact with.
in	<i>word_address</i>	device transaction type
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**Parameters**

in	<i>iface</i>	Device to interact with.
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**Parameters**

in	<i>iface</i>	Device to interact with.
in	<i>word_address</i>	device word address
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>rxlength</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

< I2C master will check ack from slave

< I2C nack value

< I2C ack value

< I2C ack value

< I2C nack value

**20.102.3.8 hal\_i2c\_release()**

```
ATCA_STATUS hal_i2c_release (
    void * hal_data )
```

manages reference count on given bus and releases resource if no more refences exist

manages reference count on given bus and releases resource if no more refernces exist

**Parameters**

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation
----	-----------------	---

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

**Parameters**

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation return ATCA_SUCCESS
in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation

**Returns**

ATCA\_SUCCESS

### 20.102.3.9 hal\_i2c\_send()

```
ATCA_STATUS hal_i2c_send (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * txdata,
    int txlength )
```

HAL implementation of I2C send.

HAL implementation of I2C send over ASF.

HAL implementation of I2C send over START.

#### Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	device transaction type
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>txlength</i>	number of bytes to send

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### Parameters

in	<i>iface</i>	instance
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>txlength</i>	number of bytes to send

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	device word address
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>txlength</i>	number of bytes to send

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

< I2C master will check ack from slave

< I2C master will check ack from slave



### 20.102.3.10 hal\_i2c\_sleep()

```
ATCA_STATUS hal_i2c_sleep (
    ATCAIface iface )
```

sleep CryptoAuth device using I2C bus

#### Parameters

in	<i>iface</i>	interface to logical device to sleep
----	--------------	--------------------------------------

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

< I2C master will check ack from slave

< I2C master will not check ack from slave

### 20.102.3.11 hal\_i2c\_wake()

```
ATCA_STATUS hal_i2c_wake (
    ATCAIface iface )
```

wake up CryptoAuth device using I2C bus

#### Parameters

in	<i>iface</i>	interface to logical device to wakeup
----	--------------	---------------------------------------

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

< I2C master will not check ack from slave

## 20.102.4 Variable Documentation

### 20.102.4.1 conf

```
i2c_config_t conf
```

### 20.102.4.2 i2c\_bus\_ref\_ct

```
int i2c_bus_ref_ct = 0
```

### 20.102.4.3 i2c\_hal\_data

```
ATCAI2CMaster_t* i2c_hal_data[2]
```

### 20.102.4.4 TAG

```
const char* TAG = "HAL_I2C"
```

## 20.103 hal\_esp32\_timer.c File Reference

```
#include "atca_hal.h"  
#include "freertos/FreeRTOS.h"  
#include "freertos/task.h"
```

### Functions

- void [ets\\_delay\\_us](#) (uint32\_t)
- void [atca\\_delay\\_ms](#) (uint32\_t msec)

### 20.103.1 Function Documentation

#### 20.103.1.1 atca\_delay\_ms()

```
void atca_delay_ms (  
    uint32_t msec )
```

#### 20.103.1.2 ets\_delay\_us()

```
void ets_delay_us (  
    uint32_t )
```

## 20.104 hal\_freertos.c File Reference

FreeRTOS Hardware/OS Abstraction Layer.

```
#include "atca_hal.h"
#include "FreeRTOS.h"
#include "semphr.h"
#include "task.h"
```

### Macros

- #define [ATCA\\_MUTEX\\_TIMEOUT](#) portMAX\_DELAY

### Functions

- void [hal\\_rtos\\_delay\\_ms](#) (uint32\_t ms)  
*Timer API implemented at the HAL level.*
- [ATCA\\_STATUS hal\\_create\\_mutex](#) (void \*\*ppMutex, char \*pName)  
*Optional hal interfaces.*
- [ATCA\\_STATUS hal\\_destroy\\_mutex](#) (void \*pMutex)
- [ATCA\\_STATUS hal\\_lock\\_mutex](#) (void \*pMutex)
- [ATCA\\_STATUS hal\\_unlock\\_mutex](#) (void \*pMutex)

### 20.104.1 Detailed Description

FreeRTOS Hardware/OS Abstraction Layer.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.104.2 Macro Definition Documentation

#### 20.104.2.1 ATCA\_MUTEX\_TIMEOUT

```
#define ATCA_MUTEX_TIMEOUT portMAX_DELAY
```

## 20.105 hal\_i2c\_harmony.c File Reference

ATCA Hardware abstraction layer for SAMD21 I2C over Harmony PLIB.

```
#include <string.h>
#include <stdio.h>
#include "cryptoauthlib.h"
```

## Functions

- [ATCA\\_STATUS hal\\_i2c\\_discover\\_buses](#) (int i2c\_buses[], int max\_buses)  
*discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge*
- [ATCA\\_STATUS hal\\_i2c\\_discover\\_devices](#) (int bus\_num, [ATCAIfaceCfg](#) cfg[], int \*found)  
*discover any CryptoAuth devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_i2c\\_init](#) (void \*hal, [ATCAIfaceCfg](#) \*cfg)  
*hal\_i2c\_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIface instances using the same bus, and you can have multiple ATCAIface instances on multiple i2c buses, so hal\_i2c\_init manages these things and ATCAIface is abstracted from the physical details.*
- [ATCA\\_STATUS hal\\_i2c\\_post\\_init](#) ([ATCAIface](#) iface)  
*HAL implementation of I2C post init.*
- [ATCA\\_STATUS hal\\_i2c\\_send](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of I2C send over START.*
- [ATCA\\_STATUS hal\\_i2c\\_receive](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)  
*HAL implementation of I2C receive function for START I2C.*
- void [change\\_i2c\\_speed](#) ([ATCAIface](#) iface, uint32\_t speed)  
*method to change the bus speed of I2C*
- [ATCA\\_STATUS hal\\_i2c\\_wake](#) ([ATCAIface](#) iface)  
*wake up CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_idle](#) ([ATCAIface](#) iface)  
*idle CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_sleep](#) ([ATCAIface](#) iface)  
*sleep CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_release](#) (void \*hal\_data)  
*manages reference count on given bus and releases resource if no more references exist*

### 20.105.1 Detailed Description

ATCA Hardware abstraction layer for SAMD21 I2C over Harmony PLIB.

This code is structured in two parts. Part 1 is the connection of the ATCA HAL API to the physical I2C implementation. Part 2 is the Harmony I2C primitives to set up the interface.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.106 hal\_i2c\_start.c File Reference

ATCA Hardware abstraction layer for SAMD21 I2C over START drivers.

```
#include <string.h>
#include <stdio.h>
#include <atmel_start.h>
#include <hal_gpio.h>
#include <hal_delay.h>
#include "hal_i2c_start.h"
#include "atca_start_config.h"
#include "atca_start_iface.h"
#include "cryptoauthlib.h"
```

## Functions

- [ATCA\\_STATUS hal\\_i2c\\_discover\\_buses](#) (int i2c\_buses[], int max\_buses)  
*discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge*
- [ATCA\\_STATUS hal\\_i2c\\_discover\\_devices](#) (int bus\_num, [ATCAIfaceCfg](#) cfg[], int \*found)  
*discover any CryptoAuth devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_i2c\\_init](#) (void \*hal, [ATCAIfaceCfg](#) \*cfg)  
*hal\_i2c\_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIface instances using the same bus, and you can have multiple ATCAIface instances on multiple i2c buses, so hal\_i2c\_init manages these things and ATCAIface is abstracted from the physical details.*
- [ATCA\\_STATUS hal\\_i2c\\_post\\_init](#) ([ATCAIface](#) iface)  
*HAL implementation of I2C post init.*
- [ATCA\\_STATUS hal\\_i2c\\_send](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of I2C send over START.*
- [ATCA\\_STATUS hal\\_i2c\\_receive](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)  
*HAL implementation of I2C receive function for START I2C.*
- [ATCA\\_STATUS hal\\_i2c\\_wake](#) ([ATCAIface](#) iface)  
*wake up CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_idle](#) ([ATCAIface](#) iface)  
*idle CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_sleep](#) ([ATCAIface](#) iface)  
*sleep CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_release](#) (void \*hal\_data)  
*manages reference count on given bus and releases resource if no more references exist*

### 20.106.1 Detailed Description

ATCA Hardware abstraction layer for SAMD21 I2C over START drivers.

This code is structured in two parts. Part 1 is the connection of the ATCA HAL API to the physical I2C implementation. Part 2 is the START I2C primitives to set up the interface.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.107 hal\_i2c\_start.h File Reference

ATCA Hardware abstraction layer for SAMD21 I2C over START drivers.

```
#include "atmel_start.h"
#include <stdlib.h>
#include "cryptoauthlib.h"
```

## Data Structures

- struct [i2c\\_start\\_instance](#)

## Typedefs

- typedef void(\* [start\\_change\\_baudrate](#)) ([ATCAIface](#) iface, uint32\_t speed)
- typedef struct [i2c\\_start\\_instance](#) [i2c\\_start\\_instance\\_t](#)

### 20.107.1 Detailed Description

ATCA Hardware abstraction layer for SAMD21 I2C over START drivers.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.108 hal\_linux.c File Reference

Timer Utility Functions for Linux.

```
#include <stdlib.h>
#include <stdint.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <fcntl.h>
#include <pthread.h>
#include <errno.h>
#include "atca_hal.h"
#include <semaphore.h>
```

## Functions

- void [hal\\_delay\\_us](#) (uint32\_t delay)  
*This function delays for a number of microseconds.*
- void [hal\\_delay\\_10us](#) (uint32\_t delay)  
*This function delays for a number of tens of microseconds.*
- void [hal\\_delay\\_ms](#) (uint32\_t delay)  
*This function delays for a number of milliseconds.*
- [ATCA\\_STATUS](#) [hal\\_create\\_mutex](#) (void \*\*ppMutex, char \*pName)  
*Optional hal interfaces.*
- [ATCA\\_STATUS](#) [hal\\_destroy\\_mutex](#) (void \*pMutex)
- [ATCA\\_STATUS](#) [hal\\_lock\\_mutex](#) (void \*pMutex)
- [ATCA\\_STATUS](#) [hal\\_unlock\\_mutex](#) (void \*pMutex)

## 20.108.1 Detailed Description

Timer Utility Functions for Linux.

Copyright

(c) 2015-2018 Microchip Technology Inc. and its subsidiaries.

## 20.109 hal\_linux\_i2c\_userspace.c File Reference

ATCA Hardware abstraction layer for Linux using I2C.

```
#include <cryptoauthlib.h>
#include <linux/i2c-dev.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include "atca_hal.h"
#include "hal_linux_i2c_userspace.h"
```

### Functions

- [ATCA\\_STATUS hal\\_i2c\\_discover\\_buses](#) (int i2c\_buses[], int max\_buses)  
*discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge*
- [ATCA\\_STATUS hal\\_i2c\\_discover\\_devices](#) (int bus\_num, [ATCAIfaceCfg](#) cfg[], int \*found)  
*discover any CryptoAuth devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_i2c\\_init](#) (void \*hal, [ATCAIfaceCfg](#) \*cfg)  
*hal\_i2c\_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIface instances using the same bus, and you can have multiple ATCAIface instances on multiple i2c buses, so hal\_i2c\_init manages these things and ATCAIface is abstracted from the physical details.*
- [ATCA\\_STATUS hal\\_i2c\\_post\\_init](#) ([ATCAIface](#) iface)  
*HAL implementation of I2C post init.*
- [ATCA\\_STATUS hal\\_i2c\\_send](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of I2C send over START.*
- [ATCA\\_STATUS hal\\_i2c\\_receive](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)  
*HAL implementation of I2C receive function for START I2C.*
- void [change\\_i2c\\_speed](#) ([ATCAIface](#) iface, uint32\_t speed)  
*method to change the bus speed of I2C*
- [ATCA\\_STATUS hal\\_i2c\\_wake](#) ([ATCAIface](#) iface)  
*wake up CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_idle](#) ([ATCAIface](#) iface)  
*idle CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_sleep](#) ([ATCAIface](#) iface)  
*sleep CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_release](#) (void \*hal\_data)  
*manages reference count on given bus and releases resource if no more references exist*

### 20.109.1 Detailed Description

ATCA Hardware abstraction layer for Linux using I2C.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.110 hal\_linux\_i2c\_userspace.h File Reference

ATCA Hardware abstraction layer for Linux using I2C.

### Data Structures

- struct [atcal2Cmaster](#)  
*this is the hal\_data for ATCA HAL for ASF SERCOM*

### Macros

- #define [MAX\\_I2C\\_BUSES](#) 2

### Typedefs

- typedef struct [atcal2Cmaster](#) [ATCAI2CMaster\\_t](#)

### 20.110.1 Detailed Description

ATCA Hardware abstraction layer for Linux using I2C.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.111 hal\_linux\_kit\_hid.c File Reference

ATCA Hardware abstraction layer for Linux using kit protocol over a USB HID device.

```
#include <libudev.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include "atca_hal.h"
#include "hal_linux_kit_hid.h"
#include "hal/kit_protocol.h"
```



## Functions

- [ATCA\\_STATUS hal\\_kit\\_hid\\_discover\\_buses](#) (int hid\_buses[], int max\_buses)  
*discover cdc buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_discover\\_devices](#) (int bus\_num, [ATCAIfaceCfg](#) cfg[], int \*found)  
*discover any CryptoAuth devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_init](#) (void \*hal, [ATCAIfaceCfg](#) \*cfg)  
*HAL implementation of Kit USB HID init.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_post\\_init](#) ([ATCAIface](#) iface)  
*HAL implementation of Kit HID post init.*
- [ATCA\\_STATUS kit\\_phy\\_send](#) ([ATCAIface](#) iface, uint8\_t \*txdata, int txlength)  
*HAL implementation of send over USB HID.*
- [ATCA\\_STATUS kit\\_phy\\_receive](#) ([ATCAIface](#) iface, uint8\_t \*rxdata, int \*rxsize)  
*HAL implementation of kit protocol send over USB HID.*
- [ATCA\\_STATUS kit\\_phy\\_num\\_found](#) (int8\_t \*num\_found)  
*Number of USB HID devices found.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_send](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of kit protocol send over USB HID.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_receive](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxsize)  
*HAL implementation of send over USB HID.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_wake](#) ([ATCAIface](#) iface)  
*Call the wake for kit protocol.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_idle](#) ([ATCAIface](#) iface)  
*Call the idle for kit protocol.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_sleep](#) ([ATCAIface](#) iface)  
*Call the sleep for kit protocol.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_release](#) (void \*hal\_data)  
*Close the physical port for HID.*

## Variables

- [atcahid\\_t\\_gHid](#)

### 20.111.1 Detailed Description

ATCA Hardware abstraction layer for Linux using kit protocol over a USB HID device.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.112 hal\_linux\_kit\_hid.h File Reference

ATCA Hardware abstraction layer for Linux using kit protocol over a USB HID device.

## Data Structures

- struct [hid\\_device](#)
- struct [atcahid](#)

## Macros

- #define [HID\\_DEVICES\\_MAX](#) 10
- #define [HID\\_PACKET\\_MAX](#) 512

## Typedefs

- typedef struct [hid\\_device](#) [hid\\_device\\_t](#)
- typedef struct [atcahid](#) [atcahid\\_t](#)

### 20.112.1 Detailed Description

ATCA Hardware abstraction layer for Linux using kit protocol over a USB HID device.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.113 hal\_linux\_spi\_userspace.c File Reference

```
#include "cryptoauthlib.h"
#include "atca_hal.h"
#include "hal_linux_spi_userspace.h"
#include <unistd.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/spi/spidev.h>
```

## Functions

- [ATCA\\_STATUS hal\\_spi\\_discover\\_buses](#) (int spi\_buses[], int max\_buses)
- [ATCA\\_STATUS hal\\_spi\\_discover\\_devices](#) (int bus\_num, [ATCAfaceCfg](#) cfg[], int \*found)  
*discover any TA100 devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_spi\\_open\\_file](#) (const char \*dev\_name, uint32\_t speed, int \*fd)  
*Open and configure the SPI device.*
- [ATCA\\_STATUS hal\\_spi\\_init](#) (void \*hal, [ATCAfaceCfg](#) \*cfg)  
*HAL implementation of SPI init.*
- [ATCA\\_STATUS hal\\_spi\\_post\\_init](#) ([ATCAface](#) iface)
- [ATCA\\_STATUS hal\\_spi\\_receive](#) ([ATCAface](#) iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*len)  
*HAL implementation of SPI receive function.*
- [ATCA\\_STATUS hal\\_spi\\_send](#) ([ATCAface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int len)  
*HAL implementation of SPI send.*
- [ATCA\\_STATUS hal\\_spi\\_wake](#) ([ATCAface](#) iface)  
*wake up CryptoAuth device using SPI bus*
- [ATCA\\_STATUS hal\\_spi\\_idle](#) ([ATCAface](#) iface)  
*idle TA100 device using SPI bus*
- [ATCA\\_STATUS hal\\_spi\\_sleep](#) ([ATCAface](#) iface)  
*sleep TA100 device using SPI bus*
- [ATCA\\_STATUS hal\\_spi\\_release](#) (void \*hal\_data)  
*manages reference count on given bus and releases resource if no more references exist*

## 20.113.1 Function Documentation

### 20.113.1.1 hal\_spi\_discover\_buses()

```
ATCA_STATUS hal_spi_discover_buses (
    int spi_buses[],
    int max_buses )
```

### 20.113.1.2 hal\_spi\_discover\_devices()

```
ATCA_STATUS hal_spi_discover_devices (
    int bus_num,
    ATCAIfaceCfg cfg[],
    int * found )
```

discover any TA100 devices on a given logical bus number

#### Parameters

in	<i>bus_num</i>	logical bus number on which to look for TA100 devices
out	<i>cfg</i>	pointer to head of an array of interface config structures which get filled in by this method
out	<i>found</i>	number of devices found on this bus

#### Returns

ATCA\_SUCCESS

### 20.113.1.3 hal\_spi\_idle()

```
ATCA_STATUS hal_spi_idle (
    ATCAIface iface )
```

idle TA100 device using SPI bus

#### Parameters

in	<i>iface</i>	interface to logical device to idle
----	--------------	-------------------------------------

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.113.1.4 hal\_spi\_init()

```
ATCA_STATUS hal_spi_init (
    void * hal,
    ATCAIfaceCfg * cfg )
```

HAL implementation of SPI init.

this implementation assumes SPI peripheral has been enabled by user . It only initialize an SPI interface using given config.

#### Parameters

in	<i>hal</i>	pointer to HAL specific data that is maintained by this HAL
in	<i>cfg</i>	pointer to HAL specific configuration data that is used to initialize this HAL

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.113.1.5 hal\_spi\_open\_file()

```
ATCA_STATUS hal_spi_open_file (
    const char * dev_name,
    uint32_t speed,
    int * fd )
```

Open and configure the SPI device.

#### Parameters

in	<i>dev_name</i>	File name in the form /dev/spidevX.Y
in	<i>speed</i>	Clock speed in Hz
out	<i>fd</i>	resulting file descriptor

### 20.113.1.6 hal\_spi\_post\_init()

```
ATCA_STATUS hal_spi_post_init (
    ATCAIface iface )
```

### 20.113.1.7 hal\_spi\_receive()

```
ATCA_STATUS hal_spi_receive (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * rxdata,
    uint16_t * len )
```

HAL implementation of SPI receive function.

#### Parameters

in	<i>iface</i>	Device to interact with.
in	<i>word_address</i>	device transaction type
out	<i>rxdata</i>	Data received will be returned here.
in, out	<i>len</i>	As input, the size of the rxdata buffer. As output, the number of bytes received.

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.113.1.8 hal\_spi\_release()

```
ATCA_STATUS hal_spi_release (
    void * hal_data )
```

manages reference count on given bus and releases resource if no more references exist

#### Parameters

in	<i>hal_data</i>	- opaque pointer to hal data structure - known only to the HAL implementation
----	-----------------	---

#### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.113.1.9 hal\_spi\_send()

```
ATCA_STATUS hal_spi_send (
    ATCAIface iface,
    uint8_t word_address,
    uint8_t * txdata,
    int len )
```

HAL implementation of SPI send.

### Parameters

in	<i>iface</i>	instance
in	<i>word_address</i>	transaction type
in	<i>txdata</i>	data to be send to device
in	<i>txdata</i>	pointer to space to bytes to send
in	<i>len</i>	number of bytes to send

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.113.1.10 hal\_spi\_sleep()

```
ATCA_STATUS hal_spi_sleep (  
    ATCAIface iface )
```

sleep TA100 device using SPI bus

### Parameters

in	<i>iface</i>	interface to logical device to sleep
----	--------------	--------------------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.113.1.11 hal\_spi\_wake()

```
ATCA_STATUS hal_spi_wake (  
    ATCAIface iface )
```

wake up CryptoAuth device using SPI bus

### Parameters

in	<i>iface</i>	interface to logical device to wakeup
----	--------------	---------------------------------------

**Returns**

ATCA\_SUCCESS on success, otherwise an error code.

## 20.114 hal\_linux\_spi\_userspace.h File Reference

ATCA Hardware abstraction layer for Linux using SPI.

**Data Structures**

- struct [atcaSPImaster](#)

**Macros**

- #define [MAX\\_SPI\\_BUSES](#) 2

**Typedefs**

- typedef struct [atcaSPImaster](#) [ATCASPIMaster\\_t](#)

### 20.114.1 Detailed Description

ATCA Hardware abstraction layer for Linux using SPI.

**Copyright**

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.115 hal\_sam0\_i2c\_asf.c File Reference

ATCA Hardware abstraction layer for SAMD21 I2C over ASF drivers.

```
#include <asf.h>
#include <string.h>
#include <stdio.h>
#include "hal_sam0_i2c_asf.h"
#include "cryptoauthlib.h"
```

## Functions

- [ATCA\\_STATUS hal\\_i2c\\_discover\\_buses](#) (int i2c\_buses[], int max\_buses)  
*discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge*
- [ATCA\\_STATUS hal\\_i2c\\_discover\\_devices](#) (int bus\_num, [ATCAIfaceCfg](#) cfg[], int \*found)  
*discover any CryptoAuth devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_i2c\\_init](#) (void \*hal, [ATCAIfaceCfg](#) \*cfg)  
*hal\_i2c\_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIface instances using the same bus, and you can have multiple ATCAIface instances on multiple i2c buses, so hal\_i2c\_init manages these things and ATCAIface is abstracted from the physical details.*
- [ATCA\\_STATUS hal\\_i2c\\_post\\_init](#) ([ATCAIface](#) iface)  
*HAL implementation of I2C post init.*
- [ATCA\\_STATUS hal\\_i2c\\_send](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of I2C send over START.*
- [ATCA\\_STATUS hal\\_i2c\\_receive](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)  
*HAL implementation of I2C receive function for START I2C.*
- [ATCA\\_STATUS hal\\_i2c\\_wake](#) ([ATCAIface](#) iface)  
*wake up CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_idle](#) ([ATCAIface](#) iface)  
*idle CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_sleep](#) ([ATCAIface](#) iface)  
*sleep CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_release](#) (void \*hal\_data)  
*manages reference count on given bus and releases resource if no more references exist*

### 20.115.1 Detailed Description

ATCA Hardware abstraction layer for SAMD21 I2C over ASF drivers.

This code is structured in two parts. Part 1 is the connection of the ATCA HAL API to the physical I2C implementation. Part 2 is the ASF I2C primitives to set up the interface.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.116 hal\_sam0\_i2c\_asf.h File Reference

ATCA Hardware abstraction layer for SAMD21 I2C over ASF drivers.

```
#include <asf.h>
#include "cryptoauthlib.h"
```

## Data Structures

- struct [i2c\\_sam0\\_instance](#)



## Typedefs

- typedef void(\* [sam0\\_change\\_baudrate](#)) ([ATCAIface](#) iface, uint32\_t speed)
- typedef struct [i2c\\_sam0\\_instance](#) [i2c\\_sam0\\_instance\\_t](#)

### 20.116.1 Detailed Description

ATCA Hardware abstraction layer for SAMD21 I2C over ASF drivers.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.116.2 Typedef Documentation

#### 20.116.2.1 [i2c\\_sam0\\_instance\\_t](#)

```
typedef struct i2c\_sam0\_instance i2c\_sam0\_instance\_t
```

#### 20.116.2.2 [sam0\\_change\\_baudrate](#)

```
typedef void(* sam0\_change\_baudrate) (ATCAIface iface, uint32_t speed)
```

## 20.117 [hal\\_sam\\_i2c\\_asf.c](#) File Reference

ATCA Hardware abstraction layer for SAM flexcom & twi I2C over ASF drivers.

```
#include <asf.h>
#include <string.h>
#include <stdio.h>
#include "cryptoauthlib.h"
#include "hal_sam_i2c_asf.h"
```

## Functions

- [ATCA\\_STATUS hal\\_i2c\\_discover\\_buses](#) (int i2c\_buses[], int max\_buses)  
*discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge*
- [ATCA\\_STATUS hal\\_i2c\\_discover\\_devices](#) (int bus\_num, [ATCAIfaceCfg](#) cfg[], int \*found)  
*discover any CryptoAuth devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_i2c\\_init](#) (void \*hal, [ATCAIfaceCfg](#) \*cfg)  
*hal\_i2c\_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIface instances using the same bus, and you can have multiple ATCAIface instances on multiple i2c buses, so hal\_i2c\_init manages these things and ATCAIface is abstracted from the physical details.*
- [ATCA\\_STATUS hal\\_i2c\\_post\\_init](#) ([ATCAIface](#) iface)  
*HAL implementation of I2C post init.*
- [ATCA\\_STATUS hal\\_i2c\\_send](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of I2C send over START.*
- [ATCA\\_STATUS hal\\_i2c\\_receive](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)  
*HAL implementation of I2C receive function for START I2C.*
- [ATCA\\_STATUS hal\\_i2c\\_wake](#) ([ATCAIface](#) iface)  
*wake up CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_idle](#) ([ATCAIface](#) iface)  
*idle CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_sleep](#) ([ATCAIface](#) iface)  
*sleep CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_release](#) (void \*hal\_data)  
*manages reference count on given bus and releases resource if no more references exist*

### 20.117.1 Detailed Description

ATCA Hardware abstraction layer for SAM flexcom & twi I2C over ASF drivers.

This code is structured in two parts. Part 1 is the connection of the ATCA HAL API to the physical I2C implementation. Part 2 is the ASF I2C primitives to set up the interface.

Prerequisite: add "TWI - Two-Wire Interface (Common API) (service)" module to application in Atmel Studio

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.118 hal\_sam\_i2c\_asf.h File Reference

ATCA Hardware abstraction layer for SAMG55 I2C over ASF drivers.

```
#include <asf.h>
#include "cryptoauthlib.h"
```

## Data Structures

- struct [i2c\\_sam\\_instance](#)

## Typedefs

- typedef void(\* [sam\\_change\\_baudrate](#)) ([ATCAIface](#) iface, uint32\_t speed)
- typedef struct [i2c\\_sam\\_instance](#) [i2c\\_sam\\_instance\\_t](#)

### 20.118.1 Detailed Description

ATCA Hardware abstraction layer for SAMG55 I2C over ASF drivers.

Prerequisite: add "TWI - Two-Wire Interface (Common API) (service)" module to application in Atmel Studio

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.119 hal\_sam\_timer\_asf.c File Reference

ATCA Hardware abstraction layer for SAMD21 timer/delay over ASF drivers.

```
#include <asf.h>
#include <delay.h>
#include "atca_hal.h"
```

## Functions

- void [atca\\_delay\\_10us](#) (uint32\_t delay)  
*This function delays for a number of tens of microseconds.*
- void [atca\\_delay\\_us](#) (uint32\_t delay)  
*This function delays for a number of microseconds.*
- void [atca\\_delay\\_ms](#) (uint32\_t ms)  
*Timer API for legacy implementations.*

### 20.119.1 Detailed Description

ATCA Hardware abstraction layer for SAMD21 timer/delay over ASF drivers.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.120 hal\_spi\_harmony.c File Reference

ATCA Hardware abstraction layer for SPI over Harmony PLIB.

```
#include <string.h>
#include <stdio.h>
#include "atca_config.h"
#include "cryptoauthlib.h"
#include "atca_hal.h"
#include "atca_device.h"
#include "definitions.h"
#include "talib/talib_defines.h"
```

### Functions

- [ATCA\\_STATUS hal\\_spi\\_discover\\_buses](#) (int spi\_buses[], int max\_buses)  
*discover spi buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge*
- [ATCA\\_STATUS hal\\_spi\\_discover\\_devices](#) (int bus\_num, [ATCAIfaceCfg](#) cfg[], int \*found)  
*discover any TA100 devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_spi\\_init](#) (void \*hal, [ATCAIfaceCfg](#) \*cfg)  
*initialize an SPI interface using given config*
- [ATCA\\_STATUS hal\\_spi\\_post\\_init](#) ([ATCAIface](#) iface)  
*HAL implementation of SPI post init.*
- [ATCA\\_STATUS hal\\_spi\\_send](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of SPI send over Harmony.*
- [ATCA\\_STATUS hal\\_spi\\_receive](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)  
*HAL implementation of SPI receive function for HARMONY SPI.*
- [ATCA\\_STATUS hal\\_spi\\_wake](#) ([ATCAIface](#) iface)  
*wake up TA100 device using SPI bus*
- [ATCA\\_STATUS hal\\_spi\\_idle](#) ([ATCAIface](#) iface)  
*idle TA100 device using SPI bus*
- [ATCA\\_STATUS hal\\_spi\\_sleep](#) ([ATCAIface](#) iface)  
*sleep TA100 device using SPI bus*
- [ATCA\\_STATUS hal\\_spi\\_release](#) (void \*hal\_data)  
*manages reference count on given bus and releases resource if no more references exist*

### 20.120.1 Detailed Description

ATCA Hardware abstraction layer for SPI over Harmony PLIB.

This code is structured in two parts. Part 1 is the connection of the ATCA HAL API to the physical SPI implementation. Part 2 is the Harmony SPI primitives to set up the interface.

Prerequisite: add SERCOM SPI Master Interrupt support to application in Mplab Harmony 3

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.121 hal\_swi\_uart.c File Reference

ATCA Hardware abstraction layer for SWI over UART drivers.

```
#include <string.h>
#include <stdio.h>
#include "atca_hal.h"
#include "hal_swi_uart.h"
#include "atca_device.h"
#include "calib/calib_execution.h"
```

### Functions

- [ATCA\\_STATUS hal\\_swi\\_discover\\_buses](#) (int swi\_buses[ ], int max\_buses)  
*discover swi buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge*
- [ATCA\\_STATUS hal\\_swi\\_discover\\_devices](#) (int bus\_num, [ATCAIfaceCfg](#) cfg[ ], int \*found)  
*discover any CryptoAuth devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_swi\\_init](#) (void \*hal, [ATCAIfaceCfg](#) \*cfg)  
*hal\_swi\_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIface instances using the same bus, and you can have multiple ATCAIface instances on multiple swi buses, so hal\_swi\_init manages these things and ATCAIface is abstracted from the physical details.*
- [ATCA\\_STATUS hal\\_swi\\_post\\_init](#) ([ATCAIface](#) iface)  
*HAL implementation of SWI post init.*
- [ATCA\\_STATUS hal\\_swi\\_send\\_flag](#) ([ATCAIface](#) iface, uint8\_t data)  
*HAL implementation of SWI send one byte over UART.*
- [ATCA\\_STATUS hal\\_swi\\_send](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of SWI send command over UART.*
- [ATCA\\_STATUS hal\\_swi\\_receive](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)  
*HAL implementation of SWI receive function over UART.*
- [ATCA\\_STATUS hal\\_swi\\_wake](#) ([ATCAIface](#) iface)  
*wake up CryptoAuth device using SWI interface*
- [ATCA\\_STATUS hal\\_swi\\_idle](#) ([ATCAIface](#) iface)  
*idle CryptoAuth device using SWI interface*
- [ATCA\\_STATUS hal\\_swi\\_sleep](#) ([ATCAIface](#) iface)  
*sleep CryptoAuth device using SWI interface*
- [ATCA\\_STATUS hal\\_swi\\_release](#) (void \*hal\_data)  
*manages reference count on given bus and releases resource if no more references exist*

### 20.121.1 Detailed Description

ATCA Hardware abstraction layer for SWI over UART drivers.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.122 hal\_swi\_uart.h File Reference

ATCA Hardware abstraction layer for SWI over UART drivers.

### Macros

- #define [SWI\\_WAKE\\_TOKEN](#) ((uint8\_t)0x00)  
*flag preceding a command*
- #define [SWI\\_FLAG\\_CMD](#) ((uint8\_t)0x77)  
*flag preceding a command*
- #define [SWI\\_FLAG\\_TX](#) ((uint8\_t)0x88)  
*flag requesting a response*
- #define [SWI\\_FLAG\\_IDLE](#) ((uint8\_t)0xBB)  
*flag requesting to go into Idle mode*
- #define [SWI\\_FLAG\\_SLEEP](#) ((uint8\_t)0xCC)  
*flag requesting to go into Sleep mode*

### Functions

- [ATCA\\_STATUS hal\\_swi\\_send\\_flag](#) ([ATCAIface](#) iface, uint8\_t data)  
*HAL implementation of SWI send one byte over UART.*

### 20.122.1 Detailed Description

ATCA Hardware abstraction layer for SWI over UART drivers.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.123 hal\_timer\_start.c File Reference

ATCA Hardware abstraction layer for SAMD21 I2C over START drivers.

```
#include <hal_delay.h>
#include "atca_hal.h"
```

### Functions

- void [atca\\_delay\\_us](#) (uint32\_t delay)  
*This function delays for a number of microseconds.*
- void [atca\\_delay\\_10us](#) (uint32\_t delay)  
*This function delays for a number of tens of microseconds.*
- void [atca\\_delay\\_ms](#) (uint32\_t ms)  
*Timer API for legacy implementations.*

### 20.123.1 Detailed Description

ATCA Hardware abstraction layer for SAMD21 I2C over START drivers.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.124 hal\_uc3\_i2c\_asf.c File Reference

ATCA Hardware abstraction layer for SAMV71 I2C over ASF drivers.

```
#include <asf.h>
#include <string.h>
#include <stdio.h>
#include "cryptoauthlib.h"
#include "hal_uc3_i2c_asf.h"
```

#### Functions

- [ATCA\\_STATUS hal\\_i2c\\_discover\\_buses](#) (int i2c\_buses[], int max\_buses)  
*discover i2c buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-prior knowledge*
- [ATCA\\_STATUS hal\\_i2c\\_discover\\_devices](#) (int bus\_num, [ATCAIfaceCfg](#) cfg[], int \*found)  
*discover any CryptoAuth devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_i2c\\_init](#) (void \*hal, [ATCAIfaceCfg](#) \*cfg)  
*hal\_i2c\_init manages requests to initialize a physical interface. it manages use counts so when an interface has released the physical layer, it will disable the interface for some other use. You can have multiple ATCAIface instances using the same bus, and you can have multiple ATCAIface instances on multiple i2c buses, so hal\_i2c\_init manages these things and ATCAIface is abstracted from the physical details.*
- [ATCA\\_STATUS hal\\_i2c\\_post\\_init](#) ([ATCAIface](#) iface)  
*HAL implementation of I2C post init.*
- [ATCA\\_STATUS hal\\_i2c\\_send](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of I2C send over START.*
- [ATCA\\_STATUS hal\\_i2c\\_receive](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxlength)  
*HAL implementation of I2C receive function for START I2C.*
- void [change\\_i2c\\_speed](#) ([ATCAIface](#) iface, uint32\_t speed)  
*method to change the bus speed of I2C*
- [ATCA\\_STATUS hal\\_i2c\\_wake](#) ([ATCAIface](#) iface)  
*wake up CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_idle](#) ([ATCAIface](#) iface)  
*idle CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_sleep](#) ([ATCAIface](#) iface)  
*sleep CryptoAuth device using I2C bus*
- [ATCA\\_STATUS hal\\_i2c\\_release](#) (void \*hal\_data)  
*manages reference count on given bus and releases resource if no more references exist*

### 20.124.1 Detailed Description

ATCA Hardware abstraction layer for SAMV71 I2C over ASF drivers.

This code is structured in two parts. Part 1 is the connection of the ATCA HAL API to the physical I2C implementation. Part 2 is the ASF I2C primitives to set up the interface.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.125 hal\_uc3\_i2c\_asf.h File Reference

ATCA Hardware abstraction layer for SAMV71 I2C over ASF drivers.

```
#include <asf.h>
#include "twi.h"
```

### Data Structures

- struct [atcal2Cmaster](#)  
*this is the hal\_data for ATCA HAL for ASF SERCOM*

### Macros

- #define [MAX\\_I2C\\_BUSES](#) 3

### Typedefs

- typedef struct [atcal2Cmaster](#) [ATCAI2CMaster\\_t](#)  
*this is the hal\_data for ATCA HAL for ASF SERCOM*

### Functions

- void [change\\_i2c\\_speed](#) ([ATCAIface](#) iface, uint32\_t speed)  
*method to change the bus speed of I2C*

### 20.125.1 Detailed Description

ATCA Hardware abstraction layer for SAMV71 I2C over ASF drivers.

Prerequisite: add SERCOM I2C Master Polled support to application in Atmel Studio

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.



## 20.126 hal\_uc3\_timer\_asf.c File Reference

ATCA Hardware abstraction layer for SAM4S I2C over ASF drivers.

```
#include <asf.h>
#include <delay.h>
#include "atca_hal.h"
```

### Functions

- void [atca\\_delay\\_us](#) (uint32\_t delay)  
*This function delays for a number of microseconds.*
- void [atca\\_delay\\_10us](#) (uint32\_t delay)  
*This function delays for a number of tens of microseconds.*
- void [atca\\_delay\\_ms](#) (uint32\_t ms)  
*Timer API for legacy implementations.*

### 20.126.1 Detailed Description

ATCA Hardware abstraction layer for SAM4S I2C over ASF drivers.

Prerequisite: add "Delay routines (service)" module to application in Atmel Studio

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.127 hal\_win\_kit\_hid.c File Reference

ATCA Hardware abstraction layer for Windows using kit protocol over a USB HID device.

```
#include "atca_hal.h"
#include "hal_win_kit_hid.h"
#include "kit_protocol.h"
#include "kit_phy.h"
#include <SetupAPI.h>
#include <stdio.h>
#include <stdlib.h>
#include <tchar.h>
```

### Macros

- #define [HID\\_GUID](#) { 0x4d1e55b2, 0xf16f, 0x11cf, 0x88, 0xcb, 0x00, 0x11, 0x11, 0x00, 0x00, 0x30 }

## Functions

- [ATCA\\_STATUS hal\\_kit\\_hid\\_init](#) (void \*hal, ATCAIfaceCfg \*cfg)  
*HAL implementation of Kit USB HID init.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_discover\\_buses](#) (int hid\_buses[], int max\_buses)  
*discover cdc buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_discover\\_devices](#) (int bus\_num, ATCAIfaceCfg cfg[], int \*found)  
*discover any CryptoAuth devices on a given logical bus number*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_post\\_init](#) (ATCAIface iface)  
*HAL implementation of Kit HID post init.*
- [ATCA\\_STATUS kit\\_phy\\_send](#) (ATCAIface iface, const char \*txdata, int txlength)  
*HAL implementation of kit protocol send .It is called by the top layer.*
- [ATCA\\_STATUS kit\\_phy\\_receive](#) (ATCAIface iface, char \*rxdata, int \*rxsize)  
*HAL implementation of kit protocol receive data.It is called by the top layer.*
- [ATCA\\_STATUS kit\\_phy\\_num\\_found](#) (int8\_t \*num\_found)  
*Number of USB HID devices found.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_send](#) (ATCAIface iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of kit protocol send over USB HID.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_receive](#) (ATCAIface iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxsize)  
*HAL implementation of send over USB HID.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_wake](#) (ATCAIface iface)  
*Call the wake for kit protocol.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_idle](#) (ATCAIface iface)  
*Call the idle for kit protocol.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_sleep](#) (ATCAIface iface)  
*Call the sleep for kit protocol.*
- [ATCA\\_STATUS hal\\_kit\\_hid\\_release](#) (void \*hal\_data)  
*Close the physical port for HID.*

## Variables

- [atcahid\\_t \\_gHid](#)

### 20.127.1 Detailed Description

ATCA Hardware abstraction layer for Windows using kit protocol over a USB HID device.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.128 hal\_win\_kit\_hid.h File Reference

ATCA Hardware abstraction layer for Windows using kit protocol over a USB HID device.

```
#include <Windows.h>
```

## Data Structures

- struct [hid\\_device](#)
- struct [atcahid](#)

## Macros

- #define [HID\\_DEVICES\\_MAX](#) 10
- #define [HID\\_PACKET\\_MAX](#) 512

## Typedefs

- typedef struct [hid\\_device](#) [hid\\_device\\_t](#)
- typedef struct [atcahid](#) [atcahid\\_t](#)

### 20.128.1 Detailed Description

ATCA Hardware abstraction layer for Windows using kit protocol over a USB HID device.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.129 hal\_windows.c File Reference

ATCA Hardware abstraction layer for windows timer functions.

```
#include <windows.h>
#include <math.h>
#include "atca_hal.h"
```

## Functions

- void [hal\\_delay\\_us](#) (uint32\_t delay)  
*This function delays for a number of microseconds.*
- void [hal\\_delay\\_10us](#) (uint32\_t delay)  
*This function delays for a number of tens of microseconds.*
- void [hal\\_delay\\_ms](#) (uint32\_t delay)  
*This function delays for a number of milliseconds.*
- [ATCA\\_STATUS](#) [hal\\_create\\_mutex](#) (void \*\*ppMutex, char \*pName)  
*Optional hal interfaces.*
- [ATCA\\_STATUS](#) [hal\\_destroy\\_mutex](#) (void \*pMutex)
- [ATCA\\_STATUS](#) [hal\\_lock\\_mutex](#) (void \*pMutex)
- [ATCA\\_STATUS](#) [hal\\_unlock\\_mutex](#) (void \*pMutex)

### 20.129.1 Detailed Description

ATCA Hardware abstraction layer for windows timer functions.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.130 io\_protection\_key.h File Reference

Provides required interface to access IO protection key.

```
#include "atca_status.h"
```

### Functions

- [ATCA\\_STATUS io\\_protection\\_get\\_key](#) (uint8\_t \*io\_key)
- [ATCA\\_STATUS io\\_protection\\_set\\_key](#) (uint8\_t \*io\_key)

### 20.130.1 Detailed Description

Provides required interface to access IO protection key.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.130.2 Function Documentation

#### 20.130.2.1 io\_protection\_get\_key()

```
ATCA_STATUS io_protection_get_key (  
    uint8_t * io_key )
```

#### 20.130.2.2 io\_protection\_set\_key()

```
ATCA_STATUS io_protection_set_key (  
    uint8_t * io_key )
```

## 20.131 kit\_phy.h File Reference

ATCA Hardware abstraction layer physical send & receive function definitions.

```
#include "cryptoauthlib.h"
```

### Functions

- [ATCA\\_STATUS kit\\_phy\\_num\\_found](#) (int8\_t \*num\_found)  
*Number of USB HID devices found.*
- [ATCA\\_STATUS kit\\_phy\\_send](#) (ATCAIface iface, const char \*txdata, int txlength)  
*HAL implementation of kit protocol send .It is called by the top layer.*
- [ATCA\\_STATUS kit\\_phy\\_receive](#) (ATCAIface iface, char \*rxdata, int \*rxsize)  
*HAL implementation of kit protocol receive data.It is called by the top layer.*

### 20.131.1 Detailed Description

ATCA Hardware abstraction layer physical send & receive function definitions.

This is included for kit protocol implementations. It is included in the kit protocol callback to actually send and receive bytes.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.132 kit\_protocol.c File Reference

Microchip Crypto Auth hardware interface object.

```
#include <stdlib.h>
#include <stdio.h>
#include "atca_compiler.h"
#include "kit_phy.h"
#include "kit_protocol.h"
#include "atca_helpers.h"
```

### Macros

- [#define KIT\\_MAX\\_SCAN\\_COUNT](#) 4
- [#define KIT\\_MAX\\_TX\\_BUF](#) 32

## Functions

- char \* [strnchr](#) (const char \*s, size\_t count, int c)
- const char \* [kit\\_id\\_from\\_devtype](#) (ATCADeviceType devtype)
- const char \* [kit\\_interface\\_from\\_kittype](#) (ATCAKitType kittype)
- [ATCA\\_STATUS kit\\_init](#) (ATCAIface iface)  
*HAL implementation of kit protocol init. This function calls back to the physical protocol to send the bytes.*
- [ATCA\\_STATUS kit\\_send](#) (ATCAIface iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of kit protocol send. This function calls back to the physical protocol to send the bytes.*
- [ATCA\\_STATUS kit\\_receive](#) (ATCAIface iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxsize)  
*HAL implementation to receive bytes and unwrap from kit protocol. This function calls back to the physical protocol to receive the bytes.*
- [ATCA\\_STATUS kit\\_wake](#) (ATCAIface iface)  
*Call the wake for kit protocol.*
- [ATCA\\_STATUS kit\\_idle](#) (ATCAIface iface)  
*Call the idle for kit protocol.*
- [ATCA\\_STATUS kit\\_sleep](#) (ATCAIface iface)  
*Call the sleep for kit protocol.*
- [ATCA\\_STATUS kit\\_wrap\\_cmd](#) (const uint8\_t \*txdata, int txlen, char \*pkitscmd, int \*nkitcmd, char target)  
*Wrap binary bytes in ascii kit protocol.*
- [ATCA\\_STATUS kit\\_parse\\_rsp](#) (const char \*pkitsbuf, int nkitbuf, uint8\_t \*kitstatus, uint8\_t \*rxdata, int \*datasize)  
*Parse the response ascii from the kit.*

### 20.132.1 Detailed Description

Microchip Crypto Auth hardware interface object.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.133 kit\_protocol.h File Reference

```
#include "cryptoauthlib.h"
```

## Macros

- #define [KIT\\_TX\\_WRAP\\_SIZE](#) (10)
- #define [KIT\\_MSG\\_SIZE](#) (32)
- #define [KIT\\_RX\\_WRAP\\_SIZE](#) (KIT\_MSG\_SIZE + 6)

## Functions

- [ATCA\\_STATUS kit\\_init](#) ([ATCAIface](#) iface)  
*HAL implementation of kit protocol init. This function calls back to the physical protocol to send the bytes.*
- [ATCA\\_STATUS kit\\_send](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*txdata, int txlength)  
*HAL implementation of kit protocol send. This function calls back to the physical protocol to send the bytes.*
- [ATCA\\_STATUS kit\\_receive](#) ([ATCAIface](#) iface, uint8\_t word\_address, uint8\_t \*rxdata, uint16\_t \*rxsize)  
*HAL implementation to receive bytes and unwrap from kit protocol. This function calls back to the physical protocol to receive the bytes.*
- [ATCA\\_STATUS kit\\_wrap\\_cmd](#) (const uint8\_t \*txdata, int txlen, char \*pkitscmd, int \*nkitcmd, char target)  
*Wrap binary bytes in ascii kit protocol.*
- [ATCA\\_STATUS kit\\_parse\\_rsp](#) (const char \*pkitsbuf, int nkitbuf, uint8\_t \*kitstatus, uint8\_t \*rxdata, int \*datasize)  
*Parse the response ascii from the kit.*
- [ATCA\\_STATUS kit\\_wake](#) ([ATCAIface](#) iface)  
*Call the wake for kit protocol.*
- [ATCA\\_STATUS kit\\_idle](#) ([ATCAIface](#) iface)  
*Call the idle for kit protocol.*
- [ATCA\\_STATUS kit\\_sleep](#) ([ATCAIface](#) iface)  
*Call the sleep for kit protocol.*

### 20.133.1 Detailed Description

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.134 license.txt File Reference

### Functions

- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party [software](#) (including open source software) that may accompany Microchip software. THIS [SOFTWARE](#) IS SUPPLIED BY MICROCHIP "AS IS". NO [WARRANTIES](#)
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS [INCLUDING ANY](#) IMPLIED [WARRANTIES](#) OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR [ANY INCIDENTAL](#) OR CONSEQUENTIAL COST OR EXPENSE OF [ANY](#) KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL [LIABILITY](#) ON ALL CLAIMS IN [ANY](#) WAY RELATED TO THIS [SOFTWARE](#) WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer \*Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution \*Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this [software](#) without specific prior written permission THIS [SOFTWARE](#) IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND [ANY EXPRESS](#) OR IMPLIED BUT NOT

LIMITED THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY

- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer \*Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution \*Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED BUT NOT LIMITED THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY OR CONSEQUENTIAL WHETHER IN STRICT OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE
- either version of the or (at your option) any later version. systemd is distributed in the hope that it will be useful

## Variables

- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these terms
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER EXPRESS
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR STATUTORY
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS SOFTWARE
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON INFRINGEMENT
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON MERCHANTABILITY
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT



- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY SPECIAL
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY PUNITIVE
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL LOSS
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL DAMAGE
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER CAUSED
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY LAW
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF ANY
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN

ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Ott

- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary forms
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without modification
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are met
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer \*Redistributions in binary form must reproduce the above copyright notice
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with

or without are permitted provided that the following conditions are this list of conditions and the following disclaimer \*Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution \*Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED WARRANTIES

- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer \*Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution \*Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED INCLUDING
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer \*Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution \*Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED BUT NOT LIMITED TO
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer \*Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution \*Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED BUT NOT LIMITED THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT

- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer \*Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution \*Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED BUT NOT LIMITED THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY INCIDENTAL
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer \*Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution \*Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED BUT NOT LIMITED THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY EXEMPLARY
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer \*Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution \*Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED BUT NOT LIMITED THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY OR CONSEQUENTIAL WHETHER IN CONTRACT
- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Mi-



crochip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer \*Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution \*Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED BUT NOT LIMITED THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY OR CONSEQUENTIAL WHETHER IN STRICT LIABILITY

- c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer \*Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution \*Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED BUT NOT LIMITED THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY OR CONSEQUENTIAL WHETHER IN STRICT OR EVEN IF ADVISED OF THE POSSIBILITY OF SUCH this code depends on the libudev h header file with the following [license](#)
- you can redistribute it and or modify it under the [terms](#) of the GNU Lesser General Public [License](#) as published by the Free Software [Foundation](#)
- either version of the [License](#)
- either version of the but WITHOUT ANY WARRANTY
- without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE See the GNU Lesser General Public [License](#) for more details You should have received a copy of the GNU Lesser General Public [License](#) along with [systemd](#)
- If not

## 20.134.1 Function Documentation

### 20.134.1.1 DAMAGES()

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your

responsibility to comply with third party [license terms](#) applicable to your use of third party  
WHETHER IMPLIED OR APPLY TO THIS [INCLUDING ANY IMPLIED WARRANTIES](#) OF NON AND FITNESS FOR A  
PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR [ANY INCIDENTAL](#) OR CONSEQUENTI↵  
AL COST OR EXPENSE OF [ANY](#) KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS B↵  
EEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLO↵  
WED BY MICROCHIP S TOTAL [LIABILITY](#) ON ALL CLAIMS IN [ANY](#) WAY RELATED TO THIS [SOFTWARE](#) WI↵  
LL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal  
Software All rights reserved Redistribution and use in source and binary with [or](#) without are  
permitted provided that the following conditions are this list of conditions and the following  
disclaimer\* Redistributions in binary form must reproduce the above copyright this list of  
conditions and the following disclaimer in the documentation and [or](#) other materials provided  
with the distribution\* Neither the name of Signal Software nor the names of its contributors  
may be used to endorse [or](#) promote products derived from this [software](#) without specific prior  
written permission THIS [SOFTWARE](#) IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS  
AND [ANY EXPRESS](#) OR IMPLIED BUT NOT LIMITED THE IMPLIED [WARRANTIES](#) OF [MERCHANTABILITY](#) AND FIT↵  
NESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTR↵  
IBUTORS BE LIABLE FOR [ANY](#) OR CONSEQUENTIAL DAMAGES (   
    [INCLUDING](#) ,  
    BUT NOT LIMITED TO,  
    PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;[LOSS](#) OF [USE](#),  
    DATA ,  
    OR PROFITS;OR BUSINESS *INTERRUPTION* )

### 20.134.1.2 or()

either version of the or (   
    at your *option* )

### 20.134.1.3 software()

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may  
use Microchip software and any derivatives exclusively with Microchip products It is your  
responsibility to comply with third party [license terms](#) applicable to your use of third party  
software (   
    including open source *software* )

### 20.134.1.4 TORT()

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may  
use Microchip [software](#) and any derivatives exclusively with Microchip products It is your  
responsibility to comply with third party [license terms](#) applicable to your use of third party  
WHETHER IMPLIED OR APPLY TO THIS [INCLUDING ANY IMPLIED WARRANTIES](#) OF NON AND FITNESS FOR A  
PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR [ANY INCIDENTAL](#) OR CONSEQUENTI↵  
AL COST OR EXPENSE OF [ANY](#) KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS B↵  
EEN ADVISED OF THE POSSIBILITY OR THE [DAMAGES](#) ARE FORESEEABLE TO THE FULLEST EXTENT ALLO↵  
WED BY MICROCHIP S TOTAL [LIABILITY](#) ON ALL CLAIMS IN [ANY](#) WAY RELATED TO THIS [SOFTWARE](#) WI↵  
LL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal

Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer\* Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution\* Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED BUT NOT LIMITED THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY OR CONSEQUENTIAL WHETHER IN STRICT OR TORT ( INCLUDING NEGLIGENCE OR OTHERWISE )

## 20.134.2 Variable Documentation

### 20.134.2.1 ANY

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF ANY

### 20.134.2.2 CAUSED

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER CAUSED

### 20.134.2.3 CONTRACT

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL AL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal

Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer\* Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution\* Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED BUT NOT LIMITED THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY OR CONSEQUENTIAL WHETHER IN CONTRACT

#### 20.134.2.4 DAMAGE

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer \*Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution \*Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED BUT NOT LIMITED THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY OR CONSEQUENTIAL WHETHER IN STRICT OR EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE

##### Initial value:

=====

If using the Linux HID driver (lib/hal/hal\_linux\_kit\_hid.c)

#### 20.134.2.5 DIRECT

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer\* Redistributions in binary form must reproduce the above copyright this list of



conditions and the following disclaimer in the documentation and `or` other materials provided with the distribution\* Neither the name of Signal Software nor the names of its contributors may be used to endorse `or` promote products derived from this `software` without specific prior written permission THIS `SOFTWARE` IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND `ANY EXPRESS` OR IMPLIED BUT NOT LIMITED THE IMPLIED `WARRANTIES` OF `MERCHANTABILITY` AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR `ANY` DIRECT

#### 20.134.2.6 EXEMPLARY

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip `software` and any derivatives exclusively with Microchip products It is your responsibility to comply with third party `license terms` applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED `WARRANTIES` OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR `ANY INCIDENTAL` OR CONSEQUENTIAL COST OR EXPENSE OF `ANY` KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE `DAMAGES` ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL `LIABILITY` ON ALL CLAIMS IN `ANY` WAY RELATED TO THIS `SOFTWARE` WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with `or` without are permitted provided that the following conditions are this list of conditions and the following disclaimer\* Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and `or` other materials provided with the distribution\* Neither the name of Signal Software nor the names of its contributors may be used to endorse `or` promote products derived from this `software` without specific prior written permission THIS `SOFTWARE` IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND `ANY EXPRESS` OR IMPLIED BUT NOT LIMITED THE IMPLIED `WARRANTIES` OF `MERCHANTABILITY` AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR `ANY` EXEMPLARY

#### 20.134.2.7 EXPRESS

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip `software` and any derivatives exclusively with Microchip products It is your responsibility to comply with third party `license terms` applicable to your use of third party WHETHER EXPRESS

#### 20.134.2.8 FEES

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip `software` and any derivatives exclusively with Microchip products It is your responsibility to comply with third party `license terms` applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED `WARRANTIES` OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR `ANY INCIDENTAL` OR CONSEQUENTIAL COST OR EXPENSE OF `ANY` KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE `DAMAGES` ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL `LIABILITY` ON ALL CLAIMS IN `ANY` WAY RELATED TO THIS `SOFTWARE` WILL NOT EXCEED THE AMOUNT OF FEES

### 20.134.2.9 forms

Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary forms

### 20.134.2.10 Foundation

you can redistribute it and [or](#) modify it under the [terms](#) of the GNU Lesser General Public License as published by the Free Software Foundation

### 20.134.2.11 INCIDENTAL

Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with [or](#) without are permitted provided that the following conditions are this list of conditions and the following disclaimer\* Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and [or](#) other materials provided with the distribution\* Neither the name of Signal Software nor the names of its contributors may be used to endorse [or](#) promote products derived from this [software](#) without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED BUT NOT LIMITED THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY INCIDENTAL

### 20.134.2.12 INCLUDING

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer\* Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution\* Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this [software](#) without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED INCLUDING

### 20.134.2.13 INDIRECT

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer \*Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution \*Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this [software](#) without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED BUT NOT LIMITED THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY INDIRECT

### 20.134.2.14 INFRINGEMENT

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON INFRINGEMENT

### 20.134.2.15 LAW

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY LAW

### 20.134.2.16 LIABILITY

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer\* Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution\* Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this [software](#) without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND ANY EXPRESS OR IMPLIED BUT NOT LIMITED THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY OR CONSEQUENTIAL WHETHER IN STRICT LIABILITY

### 20.134.2.17 license

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer\* Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and or other materials provided with the distribution\* Neither the name of Signal Software nor the names of its contributors may be used to endorse or promote products derived from this [software](#) without specific prior written permission THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS

AND ANY EXPRESS OR IMPLIED BUT NOT LIMITED THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY OR CONSEQUENTIAL WHETHER IN STRICT OR EVEN IF ADVISED OF THE POSSIBILITY OF SUCH this code depends on the libudev h header file with the following license

#### 20.134.2.18 License

either version of the License

#### 20.134.2.19 LOSS

Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL LOSS

#### 20.134.2.20 MERCHANTABILITY

Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON MERCHANTABILITY

#### 20.134.2.21 met

Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip software and any derivatives exclusively with Microchip products It is your responsibility to comply with third party license terms applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are met

### 20.134.2.22 modification

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without modification

### 20.134.2.23 not

If not

### 20.134.2.24 notice

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL AL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with or without are permitted provided that the following conditions are this list of conditions and the following disclaimer\* Redistributions in binary form must reproduce the above copyright notice

### 20.134.2.25 Ott

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS INCLUDING ANY IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INCIDENTAL OR CONSEQUENTIAL COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Ott

**20.134.2.26 PUNITIVE**

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS [INCLUDING ANY IMPLIED WARRANTIES](#) OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR [ANY PUNITIVE](#)

**20.134.2.27 SOFTWARE**

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS [INCLUDING ANY IMPLIED WARRANTIES](#) OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR [ANY INCIDENTAL](#) OR CONSEQUENTIAL COST OR EXPENSE OF [ANY](#) KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE [DAMAGES](#) ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL [LIABILITY](#) ON ALL CLAIMS IN [ANY](#) WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE

**Initial value:**

```
=====
If using the cross-platform HID driver (lib/hal/hal_all_platforms_kit_hidapi.c)
this code depends on the hidapi library with the following license:
Copyright (c) 2010
```

**20.134.2.28 SPECIAL**

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS [INCLUDING ANY IMPLIED WARRANTIES](#) OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR [ANY INCIDENTAL](#) OR CONSEQUENTIAL COST OR EXPENSE OF [ANY](#) KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE [DAMAGES](#) ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL [LIABILITY](#) ON ALL CLAIMS IN [ANY](#) WAY RELATED TO THIS [SOFTWARE](#) WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with [or](#) without are permitted provided that the following conditions are this list of conditions and the following disclaimer \*Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and [or](#) other materials provided with the distribution \*Neither the name of Signal Software nor the names of its contributors may be used to endorse [or](#) promote products derived from this [software](#) without specific prior written permission THIS [SOFTWARE](#) IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND [ANY EXPRESS](#) OR IMPLIED BUT NOT LIMITED THE IMPLIED [WARRANTIES](#) OF [MERCHANTABILITY](#) AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR [ANY SPECIAL](#)

### 20.134.2.29 STATUTORY

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR STATUTORY

### 20.134.2.30 systemd

without even the implied warranty of [MERCHANTABILITY](#) or FITNESS FOR A PARTICULAR PURPOSE See the GNU Lesser General Public [License](#) for more details You should have received a copy of the GNU Lesser General Public [License](#) along with systemd

### 20.134.2.31 terms

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these terms

### 20.134.2.32 TO

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS [INCLUDING ANY](#) IMPLIED [WARRANTIES](#) OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR [ANY INCIDENTAL](#) OR CONSEQUENTIAL COST OR EXPENSE OF [ANY](#) KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE [DAMAGES](#) ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL [LIABILITY](#) ON ALL CLAIMS IN [ANY](#) WAY RELATED TO THIS [SOFTWARE](#) WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with [or](#) without are permitted provided that the following conditions are this list of conditions and the following disclaimer\* Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and [or](#) other materials provided with the distribution\* Neither the name of Signal Software nor the names of its contributors may be used to endorse [or](#) promote products derived from this [software](#) without specific prior written permission THIS [SOFTWARE](#) IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND [ANY EXPRESS](#) OR IMPLIED BUT NOT LIMITED TO



### 20.134.2.33 WARRANTIES

c Microchip Technology Inc and its subsidiaries Subject to your compliance with these you may use Microchip [software](#) and any derivatives exclusively with Microchip products It is your responsibility to comply with third party [license terms](#) applicable to your use of third party WHETHER IMPLIED OR APPLY TO THIS [INCLUDING ANY](#) IMPLIED WARRANTIES OF NON AND FITNESS FOR A PARTICULAR PURPOSE IN NO EVENT WILL MICROCHIP BE LIABLE FOR [ANY INCIDENTAL](#) OR CONSEQUENTIAL COST OR EXPENSE OF [ANY](#) KIND WHATSOEVER RELATED TO THE HOWEVER EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE [DAMAGES](#) ARE FORESEEABLE TO THE FULLEST EXTENT ALLOWED BY MICROCHIP S TOTAL [LIABILITY](#) ON ALL CLAIMS IN [ANY](#) WAY RELATED TO THIS [SOFTWARE](#) WILL NOT EXCEED THE AMOUNT OF IF THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS Alan Signal Software All rights reserved Redistribution and use in source and binary with [or](#) without are permitted provided that the following conditions are this list of conditions and the following disclaimer\* Redistributions in binary form must reproduce the above copyright this list of conditions and the following disclaimer in the documentation and [or](#) other materials provided with the distribution\* Neither the name of Signal Software nor the names of its contributors may be used to endorse [or](#) promote products derived from this [software](#) without specific prior written permission THIS [SOFTWARE](#) IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS AS IS AND [ANY EXPRESS](#) OR IMPLIED WARRANTIES

### 20.134.2.34 WARRANTY

either version of the but WITHOUT [ANY](#) WARRANTY

## 20.135 pkcs11.h File Reference

```
#include "pkcs11t.h"
#include "pkcs11f.h"
```

### Data Structures

- struct [CK\\_FUNCTION\\_LIST](#)

### Macros

- `#define __PASTE(x, y) x ## y`
- `#define CK_NEED_ARG_LIST 1`
- `#define CK_PKCS11_FUNCTION_INFO(name) extern CK_DECLARE_FUNCTION(CK_RV, name)`
- `#define CK_NEED_ARG_LIST 1`
- `#define CK_PKCS11_FUNCTION_INFO(name) typedef CK_DECLARE_FUNCTION_POINTER (CK_RV, __PASTE (CK_, name))`
- `#define CK_PKCS11_FUNCTION_INFO(name) __PASTE(CK_, name) name;`

### 20.135.1 Macro Definition Documentation

### 20.135.1.1 \_\_PASTE

```
#define __PASTE(  
    x,  
    y ) x ## y
```

### 20.135.1.2 CK\_NEED\_ARG\_LIST [1/2]

```
#define CK_NEED_ARG_LIST 1
```

### 20.135.1.3 CK\_NEED\_ARG\_LIST [2/2]

```
#define CK_NEED_ARG_LIST 1
```

### 20.135.1.4 CK\_PKCS11\_FUNCTION\_INFO [1/3]

```
#define CK_PKCS11_FUNCTION_INFO(  
    name ) extern CK\_DECLARE\_FUNCTION(CK_RV, name)
```

### 20.135.1.5 CK\_PKCS11\_FUNCTION\_INFO [2/3]

```
#define CK_PKCS11_FUNCTION_INFO(  
    name ) typedef CK\_DECLARE\_FUNCTION\_POINTER (CK_RV, \_\_PASTE (CK_, name))
```

### 20.135.1.6 CK\_PKCS11\_FUNCTION\_INFO [3/3]

```
#define CK_PKCS11_FUNCTION_INFO(  
    name ) \_\_PASTE(CK_, name) name;
```

## 20.136 pkcs11\_attrib.c File Reference

PKCS11 Library Object Attributes Handling.

```
#include "pkcs11_config.h"  
#include "pkcs11_attrib.h"  
#include "cryptoauthlib.h"
```

## Functions

- [CK\\_RV pkcs11\\_attr\\_fill](#) ([CK\\_ATTRIBUTE\\_PTR](#) pAttribute, const [CK\\_VOID\\_PTR](#) pData, const [CK\\_ULONG](#) ulSize)  
*Perform the nessasary checks and copy data into an attribute structure.*
- [CK\\_RV pkcs11\\_attr\\_value](#) ([CK\\_ATTRIBUTE\\_PTR](#) pAttribute, const [CK\\_ULONG](#) ulValue, const [CK\\_ULONG](#) ulSize)  
*Helper function to write a numerical value to an attribute buffer.*
- [CK\\_RV pkcs11\\_attr\\_false](#) (const [CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_attr\\_true](#) (const [CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_attr\\_empty](#) (const [CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)

### 20.136.1 Detailed Description

PKCS11 Library Object Attributes Handling.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.137 pkcs11\_attr.h File Reference

PKCS11 Library Object Attribute Handling.

```
#include "cryptoki.h"
```

## Data Structures

- [struct \\_pkcs11\\_attr\\_model](#)

## Typedefs

- typedef [CK\\_RV](#)(\* [attr\\_f](#)) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- typedef struct [\\_pkcs11\\_attr\\_model](#) [pkcs11\\_attr\\_model](#)
- typedef struct [\\_pkcs11\\_attr\\_model](#) \* [pkcs11\\_attr\\_model\\_ptr](#)

## Functions

- [CK\\_RV pkcs11\\_attr\\_fill](#) ([CK\\_ATTRIBUTE\\_PTR](#) pAttribute, const [CK\\_VOID\\_PTR](#) pData, const [CK\\_ULONG](#) ulSize)  
*Perform the nessasary checks and copy data into an attribute structure.*
- [CK\\_RV pkcs11\\_attr\\_value](#) ([CK\\_ATTRIBUTE\\_PTR](#) pAttribute, const [CK\\_ULONG](#) ulValue, const [CK\\_ULONG](#) ulSize)  
*Helper function to write a numerical value to an attribute buffer.*
- [CK\\_RV pkcs11\\_attr\\_false](#) (const [CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_attr\\_true](#) (const [CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_attr\\_empty](#) (const [CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)

### 20.137.1 Detailed Description

PKCS11 Library Object Attribute Handling.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.137.2 Typedef Documentation

#### 20.137.2.1 attrib\_f

```
typedef CK_RV(* attrib_f) (CK_VOID_PTR pObject, CK_ATTRIBUTE_PTR pAttribute)
```

Populate an attribute based on the "object"

#### 20.137.2.2 pkcs11\_attrib\_model

```
typedef struct _pkcs11_attrib_model pkcs11_attrib_model
```

#### 20.137.2.3 pkcs11\_attrib\_model\_ptr

```
typedef struct _pkcs11_attrib_model * pkcs11_attrib_model_ptr
```

## 20.138 pkcs11\_cert.c File Reference

PKCS11 Library Certificate Handling.

```
#include "cryptoauthlib.h"
#include "atcacert/atcacert_def.h"
#include "atcacert/atcacert_client.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_token.h"
#include "pkcs11_cert.h"
#include "pkcs11_os.h"
#include "pkcs11_util.h"
```

## Functions

- [CK\\_RV pkcs11\\_cert\\_get\\_encoded](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_cert\\_get\\_type](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_cert\\_get\\_subject](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_cert\\_get\\_subject\\_key\\_id](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_cert\\_get\\_authority\\_key\\_id](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_cert\\_get\\_trusted\\_flag](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_cert\\_x509\\_write](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)

## Variables

- const [pkcs11\\_attr\\_model pkcs11\\_cert\\_x509public\\_attributes](#) []
- const [CK\\_ULONG pkcs11\\_cert\\_x509public\\_attributes\\_count](#) = sizeof( [pkcs11\\_cert\\_x509public\\_attributes](#) ) / sizeof( [pkcs11\\_cert\\_x509public\\_attributes](#) [0])
- const [pkcs11\\_attr\\_model pkcs11\\_cert\\_wtlspublic\\_attributes](#) []
- const [CK\\_ULONG pkcs11\\_cert\\_wtlspublic\\_attributes\\_count](#) = sizeof( [pkcs11\\_cert\\_wtlspublic\\_attributes](#) ) / sizeof( [pkcs11\\_cert\\_wtlspublic\\_attributes](#) [0])
- const [pkcs11\\_attr\\_model pkcs11\\_cert\\_x509\\_attributes](#) []
- const [CK\\_ULONG pkcs11\\_cert\\_x509\\_attributes\\_count](#) = sizeof( [pkcs11\\_cert\\_x509\\_attributes](#) ) / sizeof( [pkcs11\\_cert\\_x509\\_attributes](#) [0])

### 20.138.1 Detailed Description

PKCS11 Library Certificate Handling.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.139 pkcs11\_cert.h File Reference

PKCS11 Library Certificate Handling.

```
#include "pkcs11_object.h"
```

## Functions

- [CK\\_RV pkcs11\\_cert\\_x509\\_write](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)

## Variables

- const [pkcs11\\_attr\\_model pkcs11\\_cert\\_x509public\\_attributes](#) []
- const [CK\\_ULONG pkcs11\\_cert\\_x509public\\_attributes\\_count](#)
- const [pkcs11\\_attr\\_model pkcs11\\_cert\\_wtlspublic\\_attributes](#) []
- const [CK\\_ULONG pkcs11\\_cert\\_wtlspublic\\_attributes\\_count](#)
- const [pkcs11\\_attr\\_model pkcs11\\_cert\\_x509\\_attributes](#) []
- const [CK\\_ULONG pkcs11\\_cert\\_x509\\_attributes\\_count](#)

### 20.139.1 Detailed Description

PKCS11 Library Certificate Handling.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.140 pkcs11\_config.c File Reference

PKCS11 Library Configuration.

```
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "cryptoauthlib.h"
#include "pkcs11_slot.h"
#include "pkcs11_object.h"
#include "pkcs11_key.h"
#include "pkcs11_cert.h"
#include "pkcs11_os.h"
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
```

### Functions

- void [pkcs11\\_config\\_init\\_private](#) (pkcs11\_object\_ptr pObject, char \*label, size\_t len)
- void [pkcs11\\_config\\_init\\_public](#) (pkcs11\_object\_ptr pObject, char \*label, size\_t len)
- void [pkcs11\\_config\\_init\\_cert](#) (pkcs11\_object\_ptr pObject, char \*label, size\_t len)
- CK\_RV [pkcs11\\_config\\_cert](#) (pkcs11\_lib\_ctx\_ptr pLibCtx, pkcs11\_slot\_ctx\_ptr pSlot, pkcs11\_object\_ptr pObject, CK\_ATTRIBUTE\_PTR pLabel)
- CK\_RV [pkcs11\\_config\\_key](#) (pkcs11\_lib\_ctx\_ptr pLibCtx, pkcs11\_slot\_ctx\_ptr pSlot, pkcs11\_object\_ptr pObject, CK\_ATTRIBUTE\_PTR pLabel)
- CK\_RV [pkcs11\\_config\\_remove\\_object](#) (pkcs11\_lib\_ctx\_ptr pLibCtx, pkcs11\_slot\_ctx\_ptr pSlot, pkcs11\_object\_ptr pObject)
- CK\_RV [pkcs11\\_config\\_load\\_objects](#) (pkcs11\_slot\_ctx\_ptr slot\_ctx)
- CK\_RV [pkcs11\\_config\\_load](#) (pkcs11\_slot\_ctx\_ptr slot\_ctx)

### 20.140.1 Detailed Description

PKCS11 Library Configuration.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.141 pkcs11\_debug.c File Reference

PKCS11 Library Debugging.

```
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_os.h"
#include "atca_helpers.h"
```

### 20.141.1 Detailed Description

PKCS11 Library Debugging.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.142 pkcs11\_debug.h File Reference

PKCS11 Library Debugging.

```
#include "pkcs11_config.h"
```

### Macros

- `#define PKCS11_DEBUG_NOFILE(...)`
- `#define PKCS11_DEBUG(...)`
- `#define PKCS11_DEBUG_RETURN(x) { return x; }`
- `#define pkcs11_debug_attributes(x, y)`

### 20.142.1 Detailed Description

PKCS11 Library Debugging.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.142.2 Macro Definition Documentation

### 20.142.2.1 PKCS11\_DEBUG

```
#define PKCS11_DEBUG(  
    ... )
```

### 20.142.2.2 pkcs11\_debug\_attributes

```
#define pkcs11_debug_attributes(  
    x,  
    y )
```

### 20.142.2.3 PKCS11\_DEBUG\_NOFILE

```
#define PKCS11_DEBUG_NOFILE(  
    ... )
```

### 20.142.2.4 PKCS11\_DEBUG\_RETURN

```
#define PKCS11_DEBUG_RETURN(  
    x ) { return x; }
```

## 20.143 pkcs11\_digest.c File Reference

```
#include "pkcs11_init.h"  
#include "pkcs11_digest.h"  
#include "pkcs11_object.h"  
#include "pkcs11_session.h"  
#include "pkcs11_util.h"  
#include "cryptoauthlib.h"  
#include "crypto/atca_crypto_sw_sha2.h"
```

## Functions

- [CK\\_RV pkcs11\\_digest\\_init](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism)  
*Initializes a message-digesting operation using the specified mechanism in the specified session.*
- [CK\\_RV pkcs11\\_digest](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG](#) ulDataLen, [CK\\_BYTE\\_PTR](#) pDigest, [CK\\_ULONG\\_PTR](#) pulDigestLen)  
*Digest the specified data in a one-pass operation and return the resulting digest.*
- [CK\\_RV pkcs11\\_digest\\_update](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pPart, [CK\\_ULONG](#) ulPartLen)  
*Continues a multiple-part digesting operation.*
- [CK\\_RV pkcs11\\_digest\\_final](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pDigest, [CK\\_ULONG\\_PTR](#) pulDigestLen)  
*Finishes a multiple-part digesting operation.*



## 20.143.1 Function Documentation

### 20.143.1.1 pkcs11\_digest()

```
CK_RV pkcs11_digest (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pData,
    CK_ULONG ulDataLen,
    CK_BYTE_PTR pDigest,
    CK_ULONG_PTR pulDigestLen )
```

Digest the specified data in a one-pass operation and return the resulting digest.

### 20.143.1.2 pkcs11\_digest\_final()

```
CK_RV pkcs11_digest_final (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pDigest,
    CK_ULONG_PTR pulDigestLen )
```

Finishes a multiple-part digesting operation.

### 20.143.1.3 pkcs11\_digest\_init()

```
CK_RV pkcs11_digest_init (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism )
```

Initializes a message-digesting operation using the specified mechanism in the specified session.

### 20.143.1.4 pkcs11\_digest\_update()

```
CK_RV pkcs11_digest_update (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pPart,
    CK_ULONG ulPartLen )
```

Continues a multiple-part digesting operation.

## 20.144 pkcs11\_digest.h File Reference

PKCS11 Library Digest (SHA256) Handling.

```
#include "cryptoki.h"
```

### Functions

- [CK\\_RV pkcs11\\_digest\\_init](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism)  
*Initializes a message-digesting operation using the specified mechanism in the specified session.*
- [CK\\_RV pkcs11\\_digest](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG](#) ulDataLen, [CK\\_BYTE\\_PTR](#) pDigest, [CK\\_ULONG\\_PTR](#) pulDigestLen)  
*Digest the specified data in a one-pass operation and return the resulting digest.*
- [CK\\_RV pkcs11\\_digest\\_update](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pPart, [CK\\_ULONG](#) ulPartLen)  
*Continues a multiple-part digesting operation.*
- [CK\\_RV pkcs11\\_digest\\_final](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pDigest, [CK\\_ULONG\\_PTR](#) pulDigestLen)  
*Finishes a multiple-part digesting operation.*

### 20.144.1 Detailed Description

PKCS11 Library Digest (SHA256) Handling.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.144.2 Function Documentation

#### 20.144.2.1 pkcs11\_digest()

```
CK_RV pkcs11_digest (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pData,
    CK_ULONG ulDataLen,
    CK_BYTE_PTR pDigest,
    CK_ULONG_PTR pulDigestLen )
```

Digest the specified data in a one-pass operation and return the resulting digest.

**20.144.2.2 pkcs11\_digest\_final()**

```
CK_RV pkcs11_digest_final (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pDigest,
    CK_ULONG_PTR pulDigestLen )
```

Finishes a multiple-part digesting operation.

**20.144.2.3 pkcs11\_digest\_init()**

```
CK_RV pkcs11_digest_init (
    CK_SESSION_HANDLE hSession,
    CK_MECHANISM_PTR pMechanism )
```

Initializes a message-digesting operation using the specified mechanism in the specified session.

**20.144.2.4 pkcs11\_digest\_update()**

```
CK_RV pkcs11_digest_update (
    CK_SESSION_HANDLE hSession,
    CK_BYTE_PTR pPart,
    CK_ULONG ulPartLen )
```

Continues a multiple-part digesting operation.

**20.145 pkcs11\_find.c File Reference**

PKCS11 Library Object Find/Searching.

```
#include "cryptoauthlib.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_init.h"
#include "pkcs11_os.h"
#include "pkcs11_slot.h"
#include "pkcs11_session.h"
#include "pkcs11_find.h"
#include "pkcs11_util.h"
```

**Functions**

- [CK\\_RV pkcs11\\_find\\_init](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount)
- [CK\\_RV pkcs11\\_find\\_continue](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phObject, [CK\\_ULONG](#) ulMaxObjectCount, [CK\\_ULONG\\_PTR](#) pulObjectCount)
- [CK\\_RV pkcs11\\_find\\_finish](#) ([CK\\_SESSION\\_HANDLE](#) hSession)
- [CK\\_RV pkcs11\\_find\\_get\\_attribute](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE](#) hObject, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount)

### 20.145.1 Detailed Description

PKCS11 Library Object Find/Searching.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.146 pkcs11\_find.h File Reference

PKCS11 Library Object Find/Searching.

```
#include "cryptoki.h"
#include "pkcs11_object.h"
```

### Functions

- [CK\\_RV pkcs11\\_find\\_init](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount)
- [CK\\_RV pkcs11\\_find\\_continue](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phObject, [CK\\_ULONG](#) ulMaxObjectCount, [CK\\_ULONG\\_PTR](#) pulObjectCount)
- [CK\\_RV pkcs11\\_find\\_finish](#) ([CK\\_SESSION\\_HANDLE](#) hSession)
- [CK\\_RV pkcs11\\_find\\_get\\_attribute](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE](#) hObject, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount)

### 20.146.1 Detailed Description

PKCS11 Library Object Find/Searching.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.147 pkcs11\_info.c File Reference

PKCS11 Library Information Functions.

```
#include "cryptoauthlib.h"
#include "pkcs11_config.h"
#include "pkcs11_init.h"
#include "pkcs11_slot.h"
#include "pkcs11_session.h"
#include "pkcs11_util.h"
#include <stdio.h>
```

## Functions

- [CK\\_RV pkcs11\\_get\\_lib\\_info \(CK\\_INFO\\_PTR pInfo\)](#)

*Obtains general information about Cryptoki.*

## Variables

- const char [pkcs11\\_lib\\_manufacturer\\_id](#) [] = "Microchip Technology Inc"
- const char [pkcs11\\_lib\\_description](#) [] = "Cryptoauthlib PKCS11 Interface"

### 20.147.1 Detailed Description

PKCS11 Library Information Functions.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.148 pkcs11\_info.h File Reference

PKCS11 Library Information Functions.

```
#include "cryptoki.h"
```

## Functions

- [CK\\_RV pkcs11\\_get\\_lib\\_info \(CK\\_INFO\\_PTR pInfo\)](#)

*Obtains general information about Cryptoki.*

## Variables

- const char [pkcs11\\_lib\\_manufacturer\\_id](#) []
- const char [pkcs11\\_lib\\_description](#) []

### 20.148.1 Detailed Description

PKCS11 Library Information Functions.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.149 pkcs11\_init.c File Reference

PKCS11 Library Init/Deinit.

```
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_init.h"
#include "pkcs11_os.h"
#include "pkcs11_slot.h"
#include "pkcs11_object.h"
#include "pkcs11_session.h"
#include "cryptoauthlib.h"
```

### Functions

- [pkcs11\\_lib\\_ctx\\_ptr pkcs11\\_get\\_context](#) (void)  
*Retrieve the current library context.*
- [CK\\_RV pkcs11\\_lock\\_context](#) ([pkcs11\\_lib\\_ctx\\_ptr](#) pContext)
- [CK\\_RV pkcs11\\_unlock\\_context](#) ([pkcs11\\_lib\\_ctx\\_ptr](#) pContext)
- [CK\\_RV pkcs11\\_init\\_check](#) ([pkcs11\\_lib\\_ctx\\_ptr](#) \*ppContext, [CK\\_BBOOL](#) lock)  
*Check if the library is initialized properly.*
- [CK\\_RV pkcs11\\_init](#) ([CK\\_C\\_INITIALIZE\\_ARGS\\_PTR](#) pInitArgs)  
*Initializes the PKCS11 API Library for Cryptoauthlib.*
- [CK\\_RV pkcs11\\_deinit](#) ([CK\\_VOID\\_PTR](#) pReserved)

### 20.149.1 Detailed Description

PKCS11 Library Init/Deinit.

Copyright (c) 2017 Microchip Technology Inc. All rights reserved.

Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.150 pkcs11\_init.h File Reference

PKCS11 Library Initialization & Context.

```
#include "cryptoki.h"
#include "pkcs11_config.h"
```

### Data Structures

- [struct \\_pkcs11\\_lib\\_ctx](#)

## Typedefs

- typedef struct `_pkcs11_lib_ctx` `pkcs11_lib_ctx`
- typedef struct `_pkcs11_lib_ctx` \* `pkcs11_lib_ctx_ptr`

## Functions

- `CK_RV` `pkcs11_init` (`CK_C_INITIALIZE_ARGS_PTR` `pInitArgs`)  
*Initializes the PKCS11 API Library for Cryptoauthlib.*
- `CK_RV` `pkcs11_deinit` (`CK_VOID_PTR` `pReserved`)
- `CK_RV` `pkcs11_init_check` (`pkcs11_lib_ctx_ptr` \*`ppContext`, `CK_BBOOL` `lock`)  
*Check if the library is initialized properly.*
- `pkcs11_lib_ctx_ptr` `pkcs11_get_context` (`void`)  
*Retrieve the current library context.*
- `CK_RV` `pkcs11_lock_context` (`pkcs11_lib_ctx_ptr` `pContext`)
- `CK_RV` `pkcs11_unlock_context` (`pkcs11_lib_ctx_ptr` `pContext`)

### 20.150.1 Detailed Description

PKCS11 Library Initialization & Context.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.150.2 Typedef Documentation

#### 20.150.2.1 `pkcs11_lib_ctx`

```
typedef struct _pkcs11_lib_ctx pkcs11_lib_ctx
```

Library Context

#### 20.150.2.2 `pkcs11_lib_ctx_ptr`

```
typedef struct _pkcs11_lib_ctx * pkcs11_lib_ctx_ptr
```

## 20.151 pkcs11\_key.c File Reference

PKCS11 Library Key Object Handling.

```
#include "cryptoauthlib.h"
#include "crypto/atca_crypto_sw_sha1.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_token.h"
#include "pkcs11_attrib.h"
#include "pkcs11_key.h"
#include "pkcs11_session.h"
#include "pkcs11_slot.h"
#include "pkcs11_util.h"
#include "pkcs11_os.h"
```

### Functions

- [CK\\_RV pkcs11\\_key\\_write](#) ([CK\\_VOID\\_PTR](#) pSession, [CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_key\\_generate\\_pair](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_ATTRIBUTE\\_PTR](#) pPublicKeyTemplate, [CK\\_ULONG](#) ulPublicKeyAttributeCount, [CK\\_ATTRIBUTE\\_PTR](#) pPrivateKeyTemplate, [CK\\_ULONG](#) ulPrivateKeyAttributeCount, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phPublicKey, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phPrivateKey)
- [CK\\_RV pkcs11\\_key\\_derive](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_OBJECT\\_HANDLE](#) hBaseKey, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phKey)

### Variables

- const [pkcs11\\_attrib\\_model](#) [pkcs11\\_key\\_public\\_attributes](#) []
- const [CK\\_ULONG](#) [pkcs11\\_key\\_public\\_attributes\\_count](#) = sizeof( [pkcs11\\_key\\_public\\_attributes](#) ) / sizeof( [pkcs11\\_key\\_public\\_attributes](#) [0])
- const [pkcs11\\_attrib\\_model](#) [pkcs11\\_key\\_ec\\_public\\_attributes](#) []
- const [pkcs11\\_attrib\\_model](#) [pkcs11\\_key\\_private\\_attributes](#) []
- const [CK\\_ULONG](#) [pkcs11\\_key\\_private\\_attributes\\_count](#) = sizeof( [pkcs11\\_key\\_private\\_attributes](#) ) / sizeof( [pkcs11\\_key\\_private\\_attributes](#) [0])
- const [pkcs11\\_attrib\\_model](#) [pkcs11\\_key\\_rsa\\_private\\_attributes](#) []
- const [pkcs11\\_attrib\\_model](#) [pkcs11\\_key\\_ec\\_private\\_attributes](#) []
- const [pkcs11\\_attrib\\_model](#) [pkcs11\\_key\\_secret\\_attributes](#) []
- const [CK\\_ULONG](#) [pkcs11\\_key\\_secret\\_attributes\\_count](#) = sizeof( [pkcs11\\_key\\_secret\\_attributes](#) ) / sizeof( [pkcs11\\_key\\_secret\\_attributes](#) [0])

### 20.151.1 Detailed Description

PKCS11 Library Key Object Handling.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.



## 20.152 pkcs11\_key.h File Reference

PKCS11 Library Object Handling.

```
#include "pkcs11_object.h"
```

### Functions

- `CK_RV pkcs11_key_write` (`CK_VOID_PTR` pSession, `CK_VOID_PTR` pObject, `CK_ATTRIBUTE_PTR` pAttribute)
- `CK_RV pkcs11_key_generate_pair` (`CK_SESSION_HANDLE` hSession, `CK_MECHANISM_PTR` pMechanism, `CK_ATTRIBUTE_PTR` pPublicKeyTemplate, `CK_ULONG` ulPublicKeyAttributeCount, `CK_ATTRIBUTE_PTR` pPrivateKeyTemplate, `CK_ULONG` ulPrivateKeyAttributeCount, `CK_OBJECT_HANDLE_PTR` phPublicKey, `CK_OBJECT_HANDLE_PTR` phPrivateKey)
- `CK_RV pkcs11_key_derive` (`CK_SESSION_HANDLE` hSession, `CK_MECHANISM_PTR` pMechanism, `CK_OBJECT_HANDLE` hBaseKey, `CK_ATTRIBUTE_PTR` pTemplate, `CK_ULONG` ulCount, `CK_OBJECT_HANDLE_PTR` phKey)

### Variables

- `const pkcs11_attr_model pkcs11_key_public_attributes []`
- `const CK_ULONG pkcs11_key_public_attributes_count`
- `const pkcs11_attr_model pkcs11_key_private_attributes []`
- `const CK_ULONG pkcs11_key_private_attributes_count`
- `const pkcs11_attr_model pkcs11_key_secret_attributes []`
- `const CK_ULONG pkcs11_key_secret_attributes_count`

### 20.152.1 Detailed Description

PKCS11 Library Object Handling.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.153 pkcs11\_main.c File Reference

PKCS11 Basic library redirects based on the 2.40 specification <http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.html>.

```
#include "cryptoki.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_init.h"
#include "pkcs11_info.h"
#include "pkcs11_slot.h"
#include "pkcs11_mech.h"
#include "pkcs11_session.h"
#include "pkcs11_token.h"
#include "pkcs11_find.h"
#include "pkcs11_object.h"
#include "pkcs11_signature.h"
#include "pkcs11_digest.h"
#include "pkcs11_key.h"
```

## Functions

- [CK\\_RV C\\_Initialize](#) ([CK\\_VOID\\_PTR](#) pInitArgs)  
*Initializes Cryptoki library NOTES: If pInitArgs is a non-NULL\_PTR is must dereference to a [CK\\_C\\_INITIALIZE\\_ARGS](#) structure.*
- [CK\\_RV C\\_Finalize](#) ([CK\\_VOID\\_PTR](#) pReserved)  
*Clean up miscellaneous Cryptoki-associated resources.*
- [CK\\_RV C\\_GetInfo](#) ([CK\\_INFO\\_PTR](#) pInfo)  
*Obtains general information about Cryptoki.*
- [CK\\_RV C\\_GetFunctionList](#) ([CK\\_FUNCTION\\_LIST\\_PTR\\_PTR](#) ppFunctionList)  
*Obtains entry points of Cryptoki library functions.*
- [CK\\_RV C\\_GetSlotList](#) ([CK\\_BBOOL](#) tokenPresent, [CK\\_SLOT\\_ID\\_PTR](#) pSlotList, [CK\\_ULONG\\_PTR](#) pulCount)  
*Obtains a list of slots in the system.*
- [CK\\_RV C\\_GetSlotInfo](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_SLOT\\_INFO\\_PTR](#) pInfo)  
*Obtains information about a particular slot.*
- [CK\\_RV C\\_GetTokenInfo](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_TOKEN\\_INFO\\_PTR](#) pInfo)  
*Obtains information about a particular token.*
- [CK\\_RV C\\_GetMechanismList](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_MECHANISM\\_TYPE\\_PTR](#) pMechanismList, [CK\\_ULONG\\_PTR](#) pulCount)  
*Obtains a list of mechanisms supported by a token (in a slot)*
- [CK\\_RV C\\_GetMechanismInfo](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_MECHANISM\\_TYPE](#) type, [CK\\_MECHANISM\\_INFO\\_PTR](#) pInfo)  
*Obtains information about a particular mechanism of a token (in a slot)*
- [CK\\_RV C\\_InitToken](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_UTF8CHAR\\_PTR](#) pPin, [CK\\_ULONG](#) ulPinLen, [CK\\_UTF8CHAR\\_PTR](#) pLabel)  
*Initializes a token (in a slot)*
- [CK\\_RV C\\_InitPIN](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_UTF8CHAR\\_PTR](#) pPin, [CK\\_ULONG](#) ulPinLen)  
*Initializes the normal user's PIN.*
- [CK\\_RV C\\_SetPIN](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_UTF8CHAR\\_PTR](#) pOldPin, [CK\\_ULONG](#) ulOldLen, [CK\\_UTF8CHAR\\_PTR](#) pNewPin, [CK\\_ULONG](#) ulNewLen)  
*Modifies the PIN of the current user.*
- [CK\\_RV C\\_OpenSession](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_FLAGS](#) flags, [CK\\_VOID\\_PTR](#) pApplication, [CK\\_NOTIFY](#) notify, [CK\\_SESSION\\_HANDLE\\_PTR](#) phSession)  
*Opens a connection between an application and a particular token or sets up an application callback for token insertion.*
- [CK\\_RV C\\_CloseSession](#) ([CK\\_SESSION\\_HANDLE](#) hSession)  
*Close the given session.*
- [CK\\_RV C\\_CloseAllSessions](#) ([CK\\_SLOT\\_ID](#) slotID)  
*Close all open sessions.*
- [CK\\_RV C\\_GetSessionInfo](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_SESSION\\_INFO\\_PTR](#) pInfo)  
*Retrieve information about the specified session.*
- [CK\\_RV C\\_GetOperationState](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pOperationState, [CK\\_ULONG\\_PTR](#) pulOperationStateLen)  
*Obtains the cryptographic operations state of a session.*
- [CK\\_RV C\\_SetOperationState](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pOperationState, [CK\\_ULONG](#) ulOperationStateLen, [CK\\_OBJECT\\_HANDLE](#) hEncryptionKey, [CK\\_OBJECT\\_HANDLE](#) hAuthenticationKey)  
*Sets the cryptographic operations state of a session.*
- [CK\\_RV C\\_Login](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_USER\\_TYPE](#) userType, [CK\\_UTF8CHAR\\_PTR](#) pPin, [CK\\_ULONG](#) ulPinLen)  
*Login on the token in the specified session.*
- [CK\\_RV C\\_Logout](#) ([CK\\_SESSION\\_HANDLE](#) hSession)

- Log out of the token in the specified session.*
- **CK\_RV C\_CreateObject** (**CK\_SESSION\_HANDLE** hSession, **CK\_ATTRIBUTE\_PTR** pTemplate, **CK\_ULONG** ulCount, **CK\_OBJECT\_HANDLE\_PTR** phObject)
- Create a new object on the token in the specified session using the given attribute template.*
- **CK\_RV C\_CopyObject** (**CK\_SESSION\_HANDLE** hSession, **CK\_OBJECT\_HANDLE** hObject, **CK\_ATTRIBUTE\_PTR** pTemplate, **CK\_ULONG** ulCount, **CK\_OBJECT\_HANDLE\_PTR** phNewObject)
- Create a copy of the object with the specified handle.*
- **CK\_RV C\_DestroyObject** (**CK\_SESSION\_HANDLE** hSession, **CK\_OBJECT\_HANDLE** hObject)
- Destroy the specified object.*
- **CK\_RV C\_GetObjectSize** (**CK\_SESSION\_HANDLE** hSession, **CK\_OBJECT\_HANDLE** hObject, **CK\_ULONG\_PTR** pulSize)
- Obtains the size of an object in bytes.*
- **CK\_RV C\_GetAttributeValue** (**CK\_SESSION\_HANDLE** hSession, **CK\_OBJECT\_HANDLE** hObject, **CK\_ATTRIBUTE\_PTR** pTemplate, **CK\_ULONG** ulCount)
- Obtains an attribute value of an object.*
- **CK\_RV C\_SetAttributeValue** (**CK\_SESSION\_HANDLE** hSession, **CK\_OBJECT\_HANDLE** hObject, **CK\_ATTRIBUTE\_PTR** pTemplate, **CK\_ULONG** ulCount)
- Change or set the value of the specified attributes on the specified object.*
- **CK\_RV C\_FindObjectsInit** (**CK\_SESSION\_HANDLE** hSession, **CK\_ATTRIBUTE\_PTR** pTemplate, **CK\_ULONG** ulCount)
- Initializes an object search in the specified session using the specified attribute template as search parameters.*
- **CK\_RV C\_FindObjects** (**CK\_SESSION\_HANDLE** hSession, **CK\_OBJECT\_HANDLE\_PTR** phObject, **CK\_ULONG** ulMaxObjectCount, **CK\_ULONG\_PTR** pulObjectCount)
- Continue the search for objects in the specified session.*
- **CK\_RV C\_FindObjectsFinal** (**CK\_SESSION\_HANDLE** hSession)
- Finishes an object search operation (and cleans up)*
- **CK\_RV C\_EncryptInit** (**CK\_SESSION\_HANDLE** hSession, **CK\_MECHANISM\_PTR** pMechanism, **CK\_OBJECT\_HANDLE** hObject)
- Initializes an encryption operation using the specified mechanism and session.*
- **CK\_RV C\_Encrypt** (**CK\_SESSION\_HANDLE** hSession, **CK\_BYTE\_PTR** pData, **CK\_ULONG** ulDataLen, **CK\_BYTE\_PTR** pEncryptedData, **CK\_ULONG\_PTR** pulEncryptedDataLen)
- Perform a single operation encryption operation in the specified session.*
- **CK\_RV C\_EncryptUpdate** (**CK\_SESSION\_HANDLE** hSession, **CK\_BYTE\_PTR** pData, **CK\_ULONG** ulDataLen, **CK\_BYTE\_PTR** pEncryptedData, **CK\_ULONG\_PTR** pulEncryptedDataLen)
- Continues a multiple-part encryption operation.*
- **CK\_RV C\_EncryptFinal** (**CK\_SESSION\_HANDLE** hSession, **CK\_BYTE\_PTR** pEncryptedData, **CK\_ULONG\_PTR** pulEncryptedDataLen)
- Finishes a multiple-part encryption operation.*
- **CK\_RV C\_DecryptInit** (**CK\_SESSION\_HANDLE** hSession, **CK\_MECHANISM\_PTR** pMechanism, **CK\_OBJECT\_HANDLE** hObject)
- Initialize decryption using the specified object.*
- **CK\_RV C\_Decrypt** (**CK\_SESSION\_HANDLE** hSession, **CK\_BYTE\_PTR** pEncryptedData, **CK\_ULONG** ulEncryptedDataLen, **CK\_BYTE\_PTR** pData, **CK\_ULONG\_PTR** pulDataLen)
- Perform a single operation decryption in the given session.*
- **CK\_RV C\_DecryptUpdate** (**CK\_SESSION\_HANDLE** hSession, **CK\_BYTE\_PTR** pEncryptedData, **CK\_ULONG** ulEncryptedDataLen, **CK\_BYTE\_PTR** pData, **CK\_ULONG\_PTR** pulDataLen)
- Continues a multiple-part decryption operation.*
- **CK\_RV C\_DecryptFinal** (**CK\_SESSION\_HANDLE** hSession, **CK\_BYTE\_PTR** pData, **CK\_ULONG\_PTR** pulDataLen)
- Finishes a multiple-part decryption operation.*
- **CK\_RV C\_DigestInit** (**CK\_SESSION\_HANDLE** hSession, **CK\_MECHANISM\_PTR** pMechanism)
- Initializes a message-digesting operation using the specified mechanism in the specified session.*

- [CK\\_RV C\\_Digest](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG](#) ulDataLen, [CK\\_BYTE\\_PTR](#) pDigest, [CK\\_ULONG\\_PTR](#) pulDigestLen)  
*Digest the specified data in a one-pass operation and return the resulting digest.*
- [CK\\_RV C\\_DigestUpdate](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pPart, [CK\\_ULONG](#) ulPartLen)  
*Continues a multiple-part digesting operation.*
- [CK\\_RV C\\_DigestKey](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE](#) hObject)  
*Update a running digest operation by digesting a secret key with the specified handle.*
- [CK\\_RV C\\_DigestFinal](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pDigest, [CK\\_ULONG\\_PTR](#) pulDigestLen)  
*Finishes a multiple-part digesting operation.*
- [CK\\_RV C\\_SignInit](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_OBJECT\\_HANDLE](#) hKey)  
*Initialize a signing operation using the specified key and mechanism.*
- [CK\\_RV C\\_Sign](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG](#) ulDataLen, [CK\\_BYTE\\_PTR](#) pSignature, [CK\\_ULONG\\_PTR](#) pulSignatureLen)  
*Sign the data in a single pass operation.*
- [CK\\_RV C\\_SignUpdate](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pPart, [CK\\_ULONG](#) ulPartLen)  
*Continues a multiple-part signature operation.*
- [CK\\_RV C\\_SignFinal](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pSignature, [CK\\_ULONG\\_PTR](#) pulSignatureLen)  
*Finishes a multiple-part signature operation.*
- [CK\\_RV C\\_SignRecoverInit](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_OBJECT\\_HANDLE](#) hKey)  
*Initializes a signature operation, where the data can be recovered from the signature.*
- [CK\\_RV C\\_SignRecover](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG](#) ulDataLen, [CK\\_BYTE\\_PTR](#) pSignature, [CK\\_ULONG\\_PTR](#) pulSignatureLen)  
*Signs single-part data, where the data can be recovered from the signature.*
- [CK\\_RV C\\_VerifyInit](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_OBJECT\\_HANDLE](#) hKey)  
*Initializes a verification operation using the specified key and mechanism.*
- [CK\\_RV C\\_Verify](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG](#) ulDataLen, [CK\\_BYTE\\_PTR](#) pSignature, [CK\\_ULONG](#) ulSignatureLen)  
*Verifies a signature on single-part data.*
- [CK\\_RV C\\_VerifyUpdate](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pPart, [CK\\_ULONG](#) ulPartLen)  
*Continues a multiple-part verification operation.*
- [CK\\_RV C\\_VerifyFinal](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pSignature, [CK\\_ULONG](#) ulSignatureLen)  
*Finishes a multiple-part verification operation.*
- [CK\\_RV C\\_VerifyRecoverInit](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_OBJECT\\_HANDLE](#) hKey)  
*Initializes a verification operation where the data is recovered from the signature.*
- [CK\\_RV C\\_VerifyRecover](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pSignature, [CK\\_ULONG](#) ulSignatureLen, [CK\\_BYTE\\_PTR](#) pData, [CK\\_ULONG\\_PTR](#) pulDataLen)  
*Verifies a signature on single-part data, where the data is recovered from the signature.*
- [CK\\_RV C\\_DigestEncryptUpdate](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pPart, [CK\\_ULONG](#) ulPartLen, [CK\\_BYTE\\_PTR](#) pEncryptedPart, [CK\\_ULONG\\_PTR](#) pulEncryptedPartLen)  
*Continues simultaneous multiple-part digesting and encryption operations.*
- [CK\\_RV C\\_DecryptDigestUpdate](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pPart, [CK\\_ULONG](#) ulPartLen, [CK\\_BYTE\\_PTR](#) pDecryptedPart, [CK\\_ULONG\\_PTR](#) pulDecryptedPartLen)  
*Continues simultaneous multiple-part decryption and digesting operations.*
- [CK\\_RV C\\_SignEncryptUpdate](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pPart, [CK\\_ULONG](#) ulPartLen, [CK\\_BYTE\\_PTR](#) pEncryptedPart, [CK\\_ULONG\\_PTR](#) pulEncryptedPartLen)  
*Continues simultaneous multiple-part signature and encryption operations.*

- [CK\\_RV C\\_DecryptVerifyUpdate](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pEncryptedPart, [CK\\_ULONG](#) ulEncryptedPartLen, [CK\\_BYTE\\_PTR](#) pPart, [CK\\_ULONG\\_PTR](#) pulPartLen)  
*Continues simultaneous multiple-part decryption and verification operations.*
- [CK\\_RV C\\_GenerateKey](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phKey)  
*Generates a secret key using the specified mechanism.*
- [CK\\_RV C\\_GenerateKeyPair](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_ATTRIBUTE\\_PTR](#) pPublicKeyTemplate, [CK\\_ULONG](#) ulPublicKeyAttributeCount, [CK\\_ATTRIBUTE\\_PTR](#) pPrivateKeyTemplate, [CK\\_ULONG](#) ulPrivateKeyAttributeCount, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phPublicKey, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phPrivateKey)  
*Generates a public-key/private-key pair using the specified mechanism.*
- [CK\\_RV C\\_WrapKey](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_OBJECT\\_HANDLE](#) hWrappingKey, [CK\\_OBJECT\\_HANDLE](#) hKey, [CK\\_BYTE\\_PTR](#) pWrappedKey, [CK\\_ULONG\\_PTR](#) pulWrappedKeyLen)  
*Wraps (encrypts) the specified key using the specified wrapping key and mechanism.*
- [CK\\_RV C\\_UnwrapKey](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_OBJECT\\_HANDLE](#) hUnwrappingKey, [CK\\_BYTE\\_PTR](#) pWrappedKey, [CK\\_ULONG](#) ulWrappedKeyLen, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phKey)  
*Unwraps (decrypts) the specified key using the specified unwrapping key.*
- [CK\\_RV C\\_DeriveKey](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_MECHANISM\\_PTR](#) pMechanism, [CK\\_OBJECT\\_HANDLE](#) hBaseKey, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phKey)  
*Derive a key from the specified base key.*
- [CK\\_RV C\\_SeedRandom](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pSeed, [CK\\_ULONG](#) ulSeedLen)  
*Mixes in additional seed material to the random number generator.*
- [CK\\_RV C\\_GenerateRandom](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_BYTE\\_PTR](#) pRandomData, [CK\\_ULONG](#) ulRandomLen)  
*Generate the specified amount of random data.*
- [CK\\_RV C\\_GetFunctionStatus](#) ([CK\\_SESSION\\_HANDLE](#) hSession)  
*Legacy function - see PKCS#11 v2.40.*
- [CK\\_RV C\\_CancelFunction](#) ([CK\\_SESSION\\_HANDLE](#) hSession)  
*Legacy function.*
- [CK\\_RV C\\_WaitForSlotEvent](#) ([CK\\_FLAGS](#) flags, [CK\\_SLOT\\_ID\\_PTR](#) pSlot, [CK\\_VOID\\_PTR](#) pReserved)  
*Wait for a slot event (token insertion, removal, etc) on the specified slot to occur.*

### 20.153.1 Detailed Description

PKCS11 Basic library redirects based on the 2.40 specification <http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.html>.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.154 pkcs11\_mech.c File Reference

PKCS11 Library Mechanism Handling.

```
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_init.h"
#include "pkcs11_mech.h"
#include "pkcs11_slot.h"
#include "cryptoauthlib.h"
```

### Data Structures

- struct [\\_pkcs11\\_mech\\_table\\_e](#)

### Macros

- #define [PKCS11\\_MECH\\_ECC508\\_EC\\_CAPABILITY](#) ([CKF\\_EC\\_F\\_P](#) | [CKF\\_EC\\_NAMEDCURVE](#) | [CKF\\_EC\\_UNCOMPRESS](#))
- #define [TABLE\\_SIZE](#)(x) sizeof(x) / sizeof(x[0])

### Typedefs

- typedef struct [\\_pkcs11\\_mech\\_table\\_e](#) [pkcs11\\_mech\\_table\\_e](#)
- typedef struct [\\_pkcs11\\_mech\\_table\\_e](#) \* [pkcs11\\_mech\\_table\\_ptr](#)

### Functions

- [CK\\_RV pkcs11\\_mech\\_get\\_list](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_MECHANISM\\_TYPE\\_PTR](#) pMechanismList, [CK\\_ULONG\\_PTR](#) pulCount)
- [CK\\_RV pkcs\\_mech\\_get\\_info](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_MECHANISM\\_TYPE](#) type, [CK\\_MECHANISM\\_INFO\\_PTR](#) pInfo)

#### 20.154.1 Detailed Description

PKCS11 Library Mechanism Handling.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.155 pkcs11\_mech.h File Reference

PKCS11 Library Mechanism Handling.

```
#include "cryptoki.h"
```

### Functions

- [CK\\_RV pkcs11\\_mech\\_get\\_list](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_MECHANISM\\_TYPE\\_PTR](#) pMechanismList, [CK\\_ULONG\\_PTR](#) pulCount)
- [CK\\_RV pkcs\\_mech\\_get\\_info](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_MECHANISM\\_TYPE](#) type, [CK\\_MECHANISM\\_INFO\\_PTR](#) pInfo)

## 20.155.1 Detailed Description

PKCS11 Library Mechanism Handling.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.156 pkcs11\_object.c File Reference

PKCS11 Library Object Handling Base.

```
#include "cryptoauthlib.h"
#include "cryptoki.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_init.h"
#include "pkcs11_session.h"
#include "pkcs11_util.h"
#include "pkcs11_object.h"
#include "pkcs11_os.h"
#include "pkcs11_find.h"
#include "pkcs11_key.h"
#include "pkcs11_cert.h"
```

### Functions

- [CK\\_RV pkcs11\\_object\\_alloc](#) ([pkcs11\\_object\\_ptr](#) \*ppObject)
- \*\*
- [CK\\_RV pkcs11\\_object\\_free](#) ([pkcs11\\_object\\_ptr](#) pObject)
- [CK\\_RV pkcs11\\_object\\_check](#) ([pkcs11\\_object\\_ptr](#) \*ppObject, [CK\\_OBJECT\\_HANDLE](#) hObject)
- [CK\\_RV pkcs11\\_object\\_get\\_handle](#) ([pkcs11\\_object\\_ptr](#) pObject, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phObject)
- [CK\\_RV pkcs11\\_object\\_get\\_name](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_object\\_get\\_class](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_object\\_get\\_type](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_object\\_get\\_destroyable](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_object\\_get\\_size](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE](#) hObject, [CK\\_ULONG\\_PTR](#) pulSize)
- [CK\\_RV pkcs11\\_object\\_find](#) ([pkcs11\\_object\\_ptr](#) \*ppObject, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount)
- [CK\\_RV pkcs11\\_object\\_create](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phObject)
- *Create a new object on the token in the specified session using the given attribute template.*
- [CK\\_RV pkcs11\\_object\\_destroy](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE](#) hObject)
- *Destroy the specified object.*
- [CK\\_RV pkcs11\\_object\\_deinit](#) ([pkcs11\\_lib\\_ctx\\_ptr](#) pContext)
- [CK\\_RV pkcs11\\_object\\_load\\_handle\\_info](#) ([pkcs11\\_lib\\_ctx\\_ptr](#) pContext)

### Variables

- [pkcs11\\_object\\_cache\\_t](#) [pkcs11\\_object\\_cache](#) [PKCS11\_MAX\_OBJECTS\_ALLOWED]
- const [pkcs11\\_attr\\_model](#) [pkcs11\\_object\\_monotonic\\_attributes](#) []
- const [CK\\_ULONG](#) [pkcs11\\_object\\_monotonic\\_attributes\\_count](#) = sizeof( [pkcs11\\_object\\_monotonic\\_attributes](#) ) / sizeof( [pkcs11\\_object\\_monotonic\\_attributes](#) [0])

### 20.156.1 Detailed Description

PKCS11 Library Object Handling Base.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.157 pkcs11\_object.h File Reference

PKCS11 Library Object Handling.

```
#include "cryptoauthlib.h"
#include "cryptoki.h"
#include "pkcs11_config.h"
#include "pkcs11_attr.h"
```

### Data Structures

- struct [\\_pkcs11\\_object](#)
- struct [\\_pkcs11\\_object\\_cache\\_t](#)

### Macros

- #define [PKCS11\\_OBJECT\\_FLAG\\_DESTROYABLE](#) 0x01
- #define [PKCS11\\_OBJECT\\_FLAG\\_MODIFIABLE](#) 0x02
- #define [PKCS11\\_OBJECT\\_FLAG\\_DYNAMIC](#) 0x04
- #define [PKCS11\\_OBJECT\\_FLAG\\_SENSITIVE](#) 0x08
- #define [PKCS11\\_OBJECT\\_FLAG\\_TA\\_TYPE](#) 0x10
- #define [PKCS11\\_OBJECT\\_FLAG\\_TRUST\\_TYPE](#) 0x20

### Typedefs

- typedef struct [\\_pkcs11\\_object](#) [pkcs11\\_object](#)
- typedef struct [\\_pkcs11\\_object](#) \* [pkcs11\\_object\\_ptr](#)
- typedef struct [\\_pkcs11\\_object\\_cache\\_t](#) [pkcs11\\_object\\_cache\\_t](#)



## Functions

- [CK\\_RV pkcs11\\_object\\_alloc](#) ([pkcs11\\_object\\_ptr](#) \*ppObject)
- \*\*
- [CK\\_RV pkcs11\\_object\\_free](#) ([pkcs11\\_object\\_ptr](#) pObject)
- [CK\\_RV pkcs11\\_object\\_check](#) ([pkcs11\\_object\\_ptr](#) \*ppObject, [CK\\_OBJECT\\_HANDLE](#) handle)
- [CK\\_RV pkcs11\\_object\\_find](#) ([pkcs11\\_object\\_ptr](#) \*ppObject, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount)
- [CK\\_RV pkcs11\\_object\\_get\\_class](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_object\\_get\\_name](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_object\\_get\\_type](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_object\\_get\\_destroyable](#) ([CK\\_VOID\\_PTR](#) pObject, [CK\\_ATTRIBUTE\\_PTR](#) pAttribute)
- [CK\\_RV pkcs11\\_object\\_get\\_size](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE](#) hObject, [CK\\_ULONG\\_PTR](#) pulSize)
- [CK\\_RV pkcs11\\_object\\_get\\_handle](#) ([pkcs11\\_object\\_ptr](#) pObject, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phObject)
- [CK\\_RV pkcs11\\_object\\_create](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_ATTRIBUTE\\_PTR](#) pTemplate, [CK\\_ULONG](#) ulCount, [CK\\_OBJECT\\_HANDLE\\_PTR](#) phObject)
- *Create a new object on the token in the specified session using the given attribute template.*
- [CK\\_RV pkcs11\\_object\\_destroy](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_OBJECT\\_HANDLE](#) hObject)
- *Destroy the specified object.*
- [CK\\_RV pkcs11\\_object\\_deinit](#) ([pkcs11\\_lib\\_ctx\\_ptr](#) pContext)
- [CK\\_RV pkcs11\\_object\\_load\\_handle\\_info](#) ([pkcs11\\_lib\\_ctx\\_ptr](#) pContext)

## Variables

- [pkcs11\\_object\\_cache\\_t](#) [pkcs11\\_object\\_cache](#) []
- const [pkcs11\\_attr\\_model](#) [pkcs11\\_object\\_monotonic\\_attributes](#) []
- const [CK\\_ULONG](#) [pkcs11\\_object\\_monotonic\\_attributes\\_count](#)

### 20.157.1 Detailed Description

PKCS11 Library Object Handling.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.157.2 Macro Definition Documentation

#### 20.157.2.1 PKCS11\_OBJECT\_FLAG\_DESTROYABLE

```
#define PKCS11_OBJECT_FLAG_DESTROYABLE 0x01
```

### 20.157.2.2 PKCS11\_OBJECT\_FLAG\_DYNAMIC

```
#define PKCS11_OBJECT_FLAG_DYNAMIC 0x04
```

### 20.157.2.3 PKCS11\_OBJECT\_FLAG\_MODIFIABLE

```
#define PKCS11_OBJECT_FLAG_MODIFIABLE 0x02
```

### 20.157.2.4 PKCS11\_OBJECT\_FLAG\_SENSITIVE

```
#define PKCS11_OBJECT_FLAG_SENSITIVE 0x08
```

### 20.157.2.5 PKCS11\_OBJECT\_FLAG\_TA\_TYPE

```
#define PKCS11_OBJECT_FLAG_TA_TYPE 0x10
```

### 20.157.2.6 PKCS11\_OBJECT\_FLAG\_TRUST\_TYPE

```
#define PKCS11_OBJECT_FLAG_TRUST_TYPE 0x20
```

## 20.157.3 Typedef Documentation

### 20.157.3.1 pkcs11\_object

```
typedef struct _pkcs11_object pkcs11_object
```

### 20.157.3.2 pkcs11\_object\_cache\_t

```
typedef struct _pkcs11_object_cache_t pkcs11_object_cache_t
```

### 20.157.3.3 pkcs11\_object\_ptr

```
typedef struct _pkcs11_object * pkcs11_object_ptr
```

## 20.158 pkcs11\_os.c File Reference

PKCS11 Library Operating System Abstraction Functions.

```
#include "pkcs11_os.h"
#include "pkcs11_util.h"
```

### Functions

- [CK\\_RV pkcs11\\_os\\_create\\_mutex \(CK\\_VOID\\_PTR\\_PTR ppMutex\)](#)  
*Application callback for creating a mutex object.*
- [CK\\_RV pkcs11\\_os\\_destroy\\_mutex \(CK\\_VOID\\_PTR pMutex\)](#)
- [CK\\_RV pkcs11\\_os\\_lock\\_mutex \(CK\\_VOID\\_PTR pMutex\)](#)
- [CK\\_RV pkcs11\\_os\\_unlock\\_mutex \(CK\\_VOID\\_PTR pMutex\)](#)

### 20.158.1 Detailed Description

PKCS11 Library Operating System Abstraction Functions.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.159 pkcs11\_os.h File Reference

PKCS11 Library Operating System Abstraction.

```
#include "cryptoki.h"
#include "cryptoauthlib.h"
```

### Macros

- [#define pkcs11\\_os\\_malloc hal\\_malloc](#)
- [#define pkcs11\\_os\\_free hal\\_free](#)

### Functions

- [CK\\_RV pkcs11\\_os\\_create\\_mutex \(CK\\_VOID\\_PTR\\_PTR ppMutex\)](#)  
*Application callback for creating a mutex object.*
- [CK\\_RV pkcs11\\_os\\_destroy\\_mutex \(CK\\_VOID\\_PTR pMutex\)](#)
- [CK\\_RV pkcs11\\_os\\_lock\\_mutex \(CK\\_VOID\\_PTR pMutex\)](#)
- [CK\\_RV pkcs11\\_os\\_unlock\\_mutex \(CK\\_VOID\\_PTR pMutex\)](#)

### 20.159.1 Detailed Description

PKCS11 Library Operating System Abstraction.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.159.2 Macro Definition Documentation

#### 20.159.2.1 pkcs11\_os\_free

```
#define pkcs11_os_free hal_free
```

#### 20.159.2.2 pkcs11\_os\_malloc

```
#define pkcs11_os_malloc hal_malloc
```

## 20.160 pkcs11\_session.c File Reference

PKCS11 Library Session Handling.

```
#include "cryptoauthlib.h"
#include "host/atca_host.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_session.h"
#include "pkcs11_token.h"
#include "pkcs11_init.h"
#include "pkcs11_slot.h"
#include "pkcs11_object.h"
#include "pkcs11_os.h"
#include "pkcs11_util.h"
```

### Functions

- [pkcs11\\_session\\_ctx\\_ptr pkcs11\\_get\\_session\\_context](#) ([CK\\_SESSION\\_HANDLE](#) hSession)
- [CK\\_RV pkcs11\\_session\\_check](#) ([pkcs11\\_session\\_ctx\\_ptr](#) \*pSession, [CK\\_SESSION\\_HANDLE](#) hSession)  
*Check if the session is initialized properly.*
- [CK\\_RV pkcs11\\_session\\_open](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_FLAGS](#) flags, [CK\\_VOID\\_PTR](#) pApplication, [CK\\_NOTIFY](#) notify, [CK\\_SESSION\\_HANDLE\\_PTR](#) phSession)
- [CK\\_RV pkcs11\\_session\\_close](#) ([CK\\_SESSION\\_HANDLE](#) hSession)
- [CK\\_RV pkcs11\\_session\\_closeall](#) ([CK\\_SLOT\\_ID](#) slotID)  
*Close all sessions for a given slot - not actually all open sessions.*
- [CK\\_RV pkcs11\\_session\\_get\\_info](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_SESSION\\_INFO\\_PTR](#) pInfo)  
*Obtains information about a particular session.*
- [CK\\_RV pkcs11\\_session\\_login](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_USER\\_TYPE](#) userType, [CK\\_UTF8CHAR\\_PTR](#) pPin, [CK\\_ULONG](#) ulPinLen)
- [CK\\_RV pkcs11\\_session\\_logout](#) ([CK\\_SESSION\\_HANDLE](#) hSession)

## 20.160.1 Detailed Description

PKCS11 Library Session Handling.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.161 pkcs11\_session.h File Reference

PKCS11 Library Session Management & Context.

```
#include "cryptoki.h"
#include "pkcs11_config.h"
```

### Data Structures

- [struct \\_pkcs11\\_session\\_ctx](#)

### Typedefs

- `typedef struct _pkcs11_session_ctx pkcs11_session_ctx`
- `typedef struct _pkcs11_session_ctx * pkcs11_session_ctx_ptr`

### Functions

- [CK\\_RV pkcs11\\_session\\_check](#) ([pkcs11\\_session\\_ctx\\_ptr](#) \*pSession, [CK\\_SESSION\\_HANDLE](#) hSession)  
*Check if the session is initialized properly.*
- [CK\\_RV pkcs11\\_session\\_get\\_info](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_SESSION\\_INFO\\_PTR](#) pInfo)  
*Obtains information about a particular session.*
- [CK\\_RV pkcs11\\_session\\_open](#) ([CK\\_SLOT\\_ID](#) slotID, [CK\\_FLAGS](#) flags, [CK\\_VOID\\_PTR](#) pApplication, [CK\\_NOTIFY](#) notify, [CK\\_SESSION\\_HANDLE\\_PTR](#) phSession)
- [CK\\_RV pkcs11\\_session\\_close](#) ([CK\\_SESSION\\_HANDLE](#) hSession)
- [CK\\_RV pkcs11\\_session\\_closeall](#) ([CK\\_SLOT\\_ID](#) slotID)  
*Close all sessions for a given slot - not actually all open sessions.*
- [CK\\_RV pkcs11\\_session\\_login](#) ([CK\\_SESSION\\_HANDLE](#) hSession, [CK\\_USER\\_TYPE](#) userType, [CK\\_UTF8CHAR\\_PTR](#) pPin, [CK\\_ULONG](#) ulPinLen)
- [CK\\_RV pkcs11\\_session\\_logout](#) ([CK\\_SESSION\\_HANDLE](#) hSession)
- [CK\\_RV pkcs11\\_session\\_authorize](#) ([pkcs11\\_session\\_ctx\\_ptr](#) pSession, [CK\\_VOID\\_PTR](#) pObject)

### 20.161.1 Detailed Description

PKCS11 Library Session Management & Context.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.161.2 Typedef Documentation

#### 20.161.2.1 pkcs11\_session\_ctx

```
typedef struct _pkcs11_session_ctx pkcs11_session_ctx
```

Session Context

#### 20.161.2.2 pkcs11\_session\_ctx\_ptr

```
typedef struct _pkcs11_session_ctx * pkcs11_session_ctx_ptr
```

### 20.161.3 Function Documentation

#### 20.161.3.1 pkcs11\_session\_authorize()

```
CK_RV pkcs11_session_authorize (
    pkcs11_session_ctx_ptr pSession,
    CK_VOID_PTR pObject )
```

## 20.162 pkcs11\_signature.c File Reference

PKCS11 Library Sign/Verify Handling.

```
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_init.h"
#include "pkcs11_signature.h"
#include "pkcs11_object.h"
#include "pkcs11_session.h"
#include "pkcs11_util.h"
#include "cryptoauthlib.h"
#include "atcacert/atcacert_der.h"
```

## Functions

- `CK_RV pkcs11_signature_sign_init` (`CK_SESSION_HANDLE` hSession, `CK_MECHANISM_PTR` pMechanism, `CK_OBJECT_HANDLE` hKey)  
*Initialize a signing operation using the specified key and mechanism.*
- `CK_RV pkcs11_signature_sign` (`CK_SESSION_HANDLE` hSession, `CK_BYTE_PTR` pData, `CK_ULONG` ulDataLen, `CK_BYTE_PTR` pSignature, `CK_ULONG_PTR` pulSignatureLen)  
*Sign the data in a single pass operation.*
- `CK_RV pkcs11_signature_sign_continue` (`CK_SESSION_HANDLE` hSession, `CK_BYTE_PTR` pPart, `CK_ULONG` ulPartLen)  
*Continues a multiple-part signature operation.*
- `CK_RV pkcs11_signature_sign_finish` (`CK_SESSION_HANDLE` hSession, `CK_BYTE_PTR` pSignature, `CK_ULONG_PTR` pulSignatureLen)  
*Finishes a multiple-part signature operation.*
- `CK_RV pkcs11_signature_verify_init` (`CK_SESSION_HANDLE` hSession, `CK_MECHANISM_PTR` pMechanism, `CK_OBJECT_HANDLE` hKey)  
*Initializes a verification operation using the specified key and mechanism.*
- `CK_RV pkcs11_signature_verify` (`CK_SESSION_HANDLE` hSession, `CK_BYTE_PTR` pData, `CK_ULONG` ulDataLen, `CK_BYTE_PTR` pSignature, `CK_ULONG` ulSignatureLen)  
*Verifies a signature on single-part data.*
- `CK_RV pkcs11_signature_verify_continue` (`CK_SESSION_HANDLE` hSession, `CK_BYTE_PTR` pPart, `CK_ULONG` ulPartLen)  
*Continues a multiple-part verification operation.*
- `CK_RV pkcs11_signature_verify_finish` (`CK_SESSION_HANDLE` hSession, `CK_BYTE_PTR` pSignature, `CK_ULONG` ulSignatureLen)  
*Finishes a multiple-part verification operation.*

### 20.162.1 Detailed Description

PKCS11 Library Sign/Verify Handling.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.163 pkcs11\_signature.h File Reference

PKCS11 Library Sign/Verify Handling.

```
#include "cryptoki.h"
```

## Functions

- `CK_RV pkcs11_signature_sign_init` (`CK_SESSION_HANDLE` hSession, `CK_MECHANISM_PTR` pMechanism, `CK_OBJECT_HANDLE` hKey)  
*Initialize a signing operation using the specified key and mechanism.*
- `CK_RV pkcs11_signature_sign` (`CK_SESSION_HANDLE` hSession, `CK_BYTE_PTR` pData, `CK_ULONG` ulDataLen, `CK_BYTE_PTR` pSignature, `CK_ULONG_PTR` pulSignatureLen)  
*Sign the data in a single pass operation.*
- `CK_RV pkcs11_signature_sign_continue` (`CK_SESSION_HANDLE` hSession, `CK_BYTE_PTR` pPart, `CK_ULONG` ulPartLen)  
*Continues a multiple-part signature operation.*
- `CK_RV pkcs11_signature_sign_finish` (`CK_SESSION_HANDLE` hSession, `CK_BYTE_PTR` pSignature, `CK_ULONG_PTR` pulSignatureLen)  
*Finishes a multiple-part signature operation.*
- `CK_RV pkcs11_signature_verify_init` (`CK_SESSION_HANDLE` hSession, `CK_MECHANISM_PTR` pMechanism, `CK_OBJECT_HANDLE` hKey)  
*Initializes a verification operation using the specified key and mechanism.*
- `CK_RV pkcs11_signature_verify` (`CK_SESSION_HANDLE` hSession, `CK_BYTE_PTR` pData, `CK_ULONG` ulDataLen, `CK_BYTE_PTR` pSignature, `CK_ULONG` ulSignatureLen)  
*Verifies a signature on single-part data.*
- `CK_RV pkcs11_signature_verify_continue` (`CK_SESSION_HANDLE` hSession, `CK_BYTE_PTR` pPart, `CK_ULONG` ulPartLen)  
*Continues a multiple-part verification operation.*
- `CK_RV pkcs11_signature_verify_finish` (`CK_SESSION_HANDLE` hSession, `CK_BYTE_PTR` pSignature, `CK_ULONG` ulSignatureLen)  
*Finishes a multiple-part verification operation.*

### 20.163.1 Detailed Description

PKCS11 Library Sign/Verify Handling.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.164 pkcs11\_slot.c File Reference

PKCS11 Library Slot Handling.

```
#include "cryptoauthlib.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_init.h"
#include "pkcs11_slot.h"
#include "pkcs11_info.h"
#include "pkcs11_util.h"
#include "pkcs11_object.h"
#include "pkcs11_os.h"
#include <stdio.h>
```



## Functions

- [pkcs11\\_slot\\_ctx\\_ptr pkcs11\\_slot\\_get\\_context \(pkcs11\\_lib\\_ctx\\_ptr lib\\_ctx, CK\\_SLOT\\_ID slotID\)](#)  
*Retrieve the current slot context.*
- [CK\\_VOID\\_PTR pkcs11\\_slot\\_initslots \(CK\\_ULONG pulCount\)](#)
- [CK\\_RV pkcs11\\_slot\\_config \(CK\\_SLOT\\_ID slotID\)](#)
- [CK\\_RV pkcs11\\_slot\\_init \(CK\\_SLOT\\_ID slotID\)](#)
- [CK\\_RV pkcs11\\_slot\\_get\\_list \(CK\\_BBOOL tokenPresent, CK\\_SLOT\\_ID\\_PTR pSlotList, CK\\_ULONG\\_PTR pulCount\)](#)
- [CK\\_RV pkcs11\\_slot\\_get\\_info \(CK\\_SLOT\\_ID slotID, CK\\_SLOT\\_INFO\\_PTR pInfo\)](#)  
*Obtains information about a particular slot.*

### 20.164.1 Detailed Description

PKCS11 Library Slot Handling.

The nomenclature here can lead to some confusion - the pkcs11 slot is not the same as a device slot. So for example each slot defined here is a specific device (most systems would have only one). The "slots" as defined by the device specification would be enumerated separately as related to specific supported mechanisms as cryptographic "objects".

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.165 pkcs11\_slot.h File Reference

PKCS11 Library Slot Handling & Context.

```
#include "pkcs11_init.h"
#include "cryptoauthlib.h"
```

## Data Structures

- [struct \\_pkcs11\\_slot\\_ctx](#)

## Typedefs

- [typedef struct \\_pkcs11\\_slot\\_ctx pkcs11\\_slot\\_ctx](#)
- [typedef struct \\_pkcs11\\_slot\\_ctx \\* pkcs11\\_slot\\_ctx\\_ptr](#)

## Functions

- [CK\\_RV pkcs11\\_slot\\_init \(CK\\_SLOT\\_ID slotID\)](#)
- [CK\\_RV pkcs11\\_slot\\_config \(CK\\_SLOT\\_ID slotID\)](#)
- [CK\\_VOID\\_PTR pkcs11\\_slot\\_initslots \(CK\\_ULONG pulCount\)](#)
- [pkcs11\\_slot\\_ctx\\_ptr pkcs11\\_slot\\_get\\_context \(pkcs11\\_lib\\_ctx\\_ptr lib\\_ctx, CK\\_SLOT\\_ID slotID\)](#)  
*Retrieve the current slot context.*
- [CK\\_RV pkcs11\\_slot\\_get\\_list \(CK\\_BBOOL tokenPresent, CK\\_SLOT\\_ID\\_PTR pSlotList, CK\\_ULONG\\_PTR pulCount\)](#)
- [CK\\_RV pkcs11\\_slot\\_get\\_info \(CK\\_SLOT\\_ID slotID, CK\\_SLOT\\_INFO\\_PTR pInfo\)](#)  
*Obtains information about a particular slot.*

### 20.165.1 Detailed Description

PKCS11 Library Slot Handling & Context.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.165.2 Typedef Documentation

#### 20.165.2.1 pkcs11\_slot\_ctx

```
typedef struct _pkcs11_slot_ctx pkcs11_slot_ctx
```

Slot Context

#### 20.165.2.2 pkcs11\_slot\_ctx\_ptr

```
typedef struct _pkcs11_slot_ctx * pkcs11_slot_ctx_ptr
```

## 20.166 pkcs11\_token.c File Reference

PKCS11 Library Token Handling.

```
#include "cryptoauthlib.h"
#include "pkcs11_config.h"
#include "pkcs11_debug.h"
#include "pkcs11_token.h"
#include "pkcs11_slot.h"
#include "pkcs11_info.h"
#include "pkcs11_util.h"
#include "pkcs11_object.h"
#include "pkcs11_key.h"
#include "pkcs11_cert.h"
#include "pkcs11_session.h"
#include <stdio.h>
```

## Functions

- `CK_RV pkcs11_token_init` (`CK_SLOT_ID` slotID, `CK_UTF8CHAR_PTR` pPin, `CK_ULONG` ulPinLen, `CK_UTF8CHAR_PTR` pLabel)
- `CK_RV pkcs11_token_get_access_type` (`CK_VOID_PTR` pObject, `CK_ATTRIBUTE_PTR` pAttribute)
- `CK_RV pkcs11_token_get_writable` (`CK_VOID_PTR` pObject, `CK_ATTRIBUTE_PTR` pAttribute)
- `CK_RV pkcs11_token_get_storage` (`CK_VOID_PTR` pObject, `CK_ATTRIBUTE_PTR` pAttribute)
- `CK_RV pkcs11_token_get_info` (`CK_SLOT_ID` slotID, `CK_TOKEN_INFO_PTR` pInfo)  
*Obtains information about a particular token.*
- `CK_RV pkcs11_token_random` (`CK_SESSION_HANDLE` hSession, `CK_BYTE_PTR` pRandomData, `CK_ULONG` ulRandomLen)  
*Generate the specified amount of random data.*
- `CK_RV pkcs11_token_convert_pin_to_key` (const `CK_UTF8CHAR_PTR` pPin, const `CK_ULONG` ulPinLen, const `CK_UTF8CHAR_PTR` pSalt, const `CK_ULONG` ulSaltLen, `CK_BYTE_PTR` pKey, `CK_ULONG` ulKeyLen)
- `CK_RV pkcs11_token_set_pin` (`CK_SESSION_HANDLE` hSession, `CK_UTF8CHAR_PTR` pOldPin, `CK_ULONG` ulOldLen, `CK_UTF8CHAR_PTR` pNewPin, `CK_ULONG` ulNewLen)

### 20.166.1 Detailed Description

PKCS11 Library Token Handling.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.167 pkcs11\_token.h File Reference

PKCS11 Library Token Management & Context.

```
#include "pkcs11_init.h"
```

## Functions

- `CK_RV pkcs11_token_init` (`CK_SLOT_ID` slotID, `CK_UTF8CHAR_PTR` pPin, `CK_ULONG` ulPinLen, `CK_UTF8CHAR_PTR` pLabel)
- `CK_RV pkcs11_token_get_access_type` (`CK_VOID_PTR` pObject, `CK_ATTRIBUTE_PTR` pAttribute)
- `CK_RV pkcs11_token_get_writable` (`CK_VOID_PTR` pObject, `CK_ATTRIBUTE_PTR` pAttribute)
- `CK_RV pkcs11_token_get_storage` (`CK_VOID_PTR` pObject, `CK_ATTRIBUTE_PTR` pAttribute)
- `CK_RV pkcs11_token_get_info` (`CK_SLOT_ID` slotID, `CK_TOKEN_INFO_PTR` pInfo)  
*Obtains information about a particular token.*
- `CK_RV pkcs11_token_convert_pin_to_key` (const `CK_UTF8CHAR_PTR` pPin, const `CK_ULONG` ulPinLen, const `CK_UTF8CHAR_PTR` pSalt, const `CK_ULONG` ulSaltLen, `CK_BYTE_PTR` pKey, `CK_ULONG` ulKeyLen)
- `CK_RV pkcs11_token_random` (`CK_SESSION_HANDLE` hSession, `CK_BYTE_PTR` pRandomData, `CK_ULONG` ulRandomLen)  
*Generate the specified amount of random data.*
- `CK_RV pkcs11_token_set_pin` (`CK_SESSION_HANDLE` hSession, `CK_UTF8CHAR_PTR` pOldPin, `CK_ULONG` ulOldLen, `CK_UTF8CHAR_PTR` pNewPin, `CK_ULONG` ulNewLen)

### 20.167.1 Detailed Description

PKCS11 Library Token Management & Context.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.168 pkcs11\_util.c File Reference

PKCS11 Library Utility Functions.

```
#include "pkcs11_util.h"
```

### Functions

- void [pkcs11\\_util\\_escape\\_string](#) ([CK\\_UTF8CHAR\\_PTR](#) buf, [CK\\_ULONG](#) buf\_len)
- [CK\\_RV](#) [pkcs11\\_util\\_convert\\_rv](#) ([ATCA\\_STATUS](#) status)
- int [pkcs11\\_util\\_memset](#) (void \*dest, size\_t destsz, int ch, size\_t count)

### 20.168.1 Detailed Description

PKCS11 Library Utility Functions.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.169 pkcs11\_util.h File Reference

PKCS11 Library Utilities.

```
#include "pkcs11_config.h"  
#include "cryptoki.h"  
#include "cryptoauthlib.h"
```

### Macros

- #define [PKCS11\\_UTIL\\_ARRAY\\_SIZE](#)(x) sizeof(x) / sizeof(x[0])

### Functions

- void [pkcs11\\_util\\_escape\\_string](#) ([CK\\_UTF8CHAR\\_PTR](#) buf, [CK\\_ULONG](#) buf\_len)
- [CK\\_RV](#) [pkcs11\\_util\\_convert\\_rv](#) ([ATCA\\_STATUS](#) status)
- int [pkcs11\\_util\\_memset](#) (void \*dest, size\_t destsz, int ch, size\_t count)

## 20.169.1 Detailed Description

PKCS11 Library Utilities.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.169.2 Macro Definition Documentation

### 20.169.2.1 PKCS11\_UTIL\_ARRAY\_SIZE

```
#define PKCS11_UTIL_ARRAY_SIZE(  
    x ) sizeof(x) / sizeof(x[0])
```

## 20.170 pkcs11f.h File Reference

## 20.171 pkcs11t.h File Reference

### Data Structures

- struct [CK\\_VERSION](#)
- struct [CK\\_INFO](#)
- struct [CK\\_SLOT\\_INFO](#)
- struct [CK\\_TOKEN\\_INFO](#)
- struct [CK\\_SESSION\\_INFO](#)
- struct [CK\\_ATTRIBUTE](#)
- struct [CK\\_DATE](#)
- struct [CK\\_MECHANISM](#)
- struct [CK\\_MECHANISM\\_INFO](#)
- struct [CK\\_C\\_INITIALIZE\\_ARGS](#)
- struct [CK\\_RSA\\_PKCS\\_OAEP\\_PARAMS](#)
- struct [CK\\_RSA\\_PKCS\\_PSS\\_PARAMS](#)
- struct [CK\\_ECDH1\\_DERIVE\\_PARAMS](#)
- struct [CK\\_ECDH2\\_DERIVE\\_PARAMS](#)
- struct [CK\\_ECMQV\\_DERIVE\\_PARAMS](#)
- struct [CK\\_X9\\_42\\_DH1\\_DERIVE\\_PARAMS](#)
- struct [CK\\_X9\\_42\\_DH2\\_DERIVE\\_PARAMS](#)
- struct [CK\\_X9\\_42\\_MQV\\_DERIVE\\_PARAMS](#)
- struct [CK\\_KEA\\_DERIVE\\_PARAMS](#)
- struct [CK\\_RC2\\_CBC\\_PARAMS](#)
- struct [CK\\_RC2\\_MAC\\_GENERAL\\_PARAMS](#)
- struct [CK\\_RC5\\_PARAMS](#)
- struct [CK\\_RC5\\_CBC\\_PARAMS](#)
- struct [CK\\_RC5\\_MAC\\_GENERAL\\_PARAMS](#)
- struct [CK\\_DES\\_CBC\\_ENCRYPT\\_DATA\\_PARAMS](#)

- struct [CK\\_AES\\_CBC\\_ENCRYPT\\_DATA\\_PARAMS](#)
- struct [CK\\_SKIPJACK\\_PRIVATE\\_WRAP\\_PARAMS](#)
- struct [CK\\_SKIPJACK\\_RELAYX\\_PARAMS](#)
- struct [CK\\_PBE\\_PARAMS](#)
- struct [CK\\_KEY\\_WRAP\\_SET\\_OAEP\\_PARAMS](#)
- struct [CK\\_SSL3\\_RANDOM\\_DATA](#)
- struct [CK\\_SSL3\\_MASTER\\_KEY\\_DERIVE\\_PARAMS](#)
- struct [CK\\_SSL3\\_KEY\\_MAT\\_OUT](#)
- struct [CK\\_SSL3\\_KEY\\_MAT\\_PARAMS](#)
- struct [CK\\_TLS\\_PRF\\_PARAMS](#)
- struct [CK\\_WTLS\\_RANDOM\\_DATA](#)
- struct [CK\\_WTLS\\_MASTER\\_KEY\\_DERIVE\\_PARAMS](#)
- struct [CK\\_WTLS\\_PRF\\_PARAMS](#)
- struct [CK\\_WTLS\\_KEY\\_MAT\\_OUT](#)
- struct [CK\\_WTLS\\_KEY\\_MAT\\_PARAMS](#)
- struct [CK\\_CMS\\_SIG\\_PARAMS](#)
- struct [CK\\_KEY\\_DERIVATION\\_STRING\\_DATA](#)
- struct [CK\\_PKCS5\\_PBKD2\\_PARAMS](#)
- struct [CK\\_PKCS5\\_PBKD2\\_PARAMS2](#)
- struct [CK\\_OTP\\_PARAM](#)
- struct [CK\\_OTP\\_PARAMS](#)
- struct [CK\\_OTP\\_SIGNATURE\\_INFO](#)
- struct [CK\\_KIP\\_PARAMS](#)
- struct [CK\\_AES\\_CTR\\_PARAMS](#)
- struct [CK\\_GCM\\_PARAMS](#)
- struct [CK\\_CCM\\_PARAMS](#)
- struct [CK\\_AES\\_GCM\\_PARAMS](#)
- struct [CK\\_AES\\_CCM\\_PARAMS](#)
- struct [CK\\_CAMELLIA\\_CTR\\_PARAMS](#)
- struct [CK\\_CAMELLIA\\_CBC\\_ENCRYPT\\_DATA\\_PARAMS](#)
- struct [CK\\_ARIA\\_CBC\\_ENCRYPT\\_DATA\\_PARAMS](#)
- struct [CK\\_DSA\\_PARAMETER\\_GEN\\_PARAM](#)
- struct [CK\\_ECDH\\_AES\\_KEY\\_WRAP\\_PARAMS](#)
- struct [CK\\_RSA\\_AES\\_KEY\\_WRAP\\_PARAMS](#)
- struct [CK\\_TLS12\\_MASTER\\_KEY\\_DERIVE\\_PARAMS](#)
- struct [CK\\_TLS12\\_KEY\\_MAT\\_PARAMS](#)
- struct [CK\\_TLS\\_KDF\\_PARAMS](#)
- struct [CK\\_TLS\\_MAC\\_PARAMS](#)
- struct [CK\\_GOSTR3410\\_DERIVE\\_PARAMS](#)
- struct [CK\\_GOSTR3410\\_KEY\\_WRAP\\_PARAMS](#)
- struct [CK\\_SEED\\_CBC\\_ENCRYPT\\_DATA\\_PARAMS](#)

## Macros

- #define [CRYPTOKI\\_VERSION\\_MAJOR](#) 2
- #define [CRYPTOKI\\_VERSION\\_MINOR](#) 40
- #define [CRYPTOKI\\_VERSION\\_AMENDMENT](#) 0
- #define [CK\\_TRUE](#) 1
- #define [CK\\_FALSE](#) 0
- #define [FALSE](#) [CK\\_FALSE](#)
- #define [TRUE](#) [CK\\_TRUE](#)
- #define [CK\\_UNAVAILABLE\\_INFORMATION](#) (~0UL)
- #define [CK\\_EFFECTIVELY\\_INFINITE](#) 0UL
- #define [CK\\_INVALID\\_HANDLE](#) 0UL

- #define CKN\_SURRENDER 0UL
- #define CKN\_OTP\_CHANGED 1UL
- #define CKF\_TOKEN\_PRESENT 0x00000001UL /\* a token is there \*/
- #define CKF\_REMOVABLE\_DEVICE 0x00000002UL /\* removable devices\*/
- #define CKF\_HW\_SLOT 0x00000004UL /\* hardware slot \*/
- #define CKF\_RNG 0x00000001UL /\* has random # generator \*/
- #define CKF\_WRITE\_PROTECTED 0x00000002UL /\* token is write-protected \*/
- #define CKF\_LOGIN\_REQUIRED 0x00000004UL /\* user must login \*/
- #define CKF\_USER\_PIN\_INITIALIZED 0x00000008UL /\* normal user's PIN is set \*/
- #define CKF\_RESTORE\_KEY\_NOT\_NEEDED 0x00000020UL
- #define CKF\_CLOCK\_ON\_TOKEN 0x00000040UL
- #define CKF\_PROTECTED\_AUTHENTICATION\_PATH 0x00000100UL
- #define CKF\_DUAL\_CRYPTO\_OPERATIONS 0x00000200UL
- #define CKF\_TOKEN\_INITIALIZED 0x00000400UL
- #define CKF\_SECONDARY\_AUTHENTICATION 0x00000800UL
- #define CKF\_USER\_PIN\_COUNT\_LOW 0x00010000UL
- #define CKF\_USER\_PIN\_FINAL\_TRY 0x00020000UL
- #define CKF\_USER\_PIN\_LOCKED 0x00040000UL
- #define CKF\_USER\_PIN\_TO\_BE\_CHANGED 0x00080000UL
- #define CKF\_SO\_PIN\_COUNT\_LOW 0x00100000UL
- #define CKF\_SO\_PIN\_FINAL\_TRY 0x00200000UL
- #define CKF\_SO\_PIN\_LOCKED 0x00400000UL
- #define CKF\_SO\_PIN\_TO\_BE\_CHANGED 0x00800000UL
- #define CKF\_ERROR\_STATE 0x01000000UL
- #define CKU\_SO 0UL
- #define CKU\_USER 1UL
- #define CKU\_CONTEXT\_SPECIFIC 2UL
- #define CKS\_RO\_PUBLIC\_SESSION 0UL
- #define CKS\_RO\_USER\_FUNCTIONS 1UL
- #define CKS\_RW\_PUBLIC\_SESSION 2UL
- #define CKS\_RW\_USER\_FUNCTIONS 3UL
- #define CKS\_RW\_SO\_FUNCTIONS 4UL
- #define CKF\_RW\_SESSION 0x00000002UL /\* session is r/w \*/
- #define CKF\_SERIAL\_SESSION 0x00000004UL /\* no parallel \*/
- #define CKO\_DATA 0x00000000UL
- #define CKO\_CERTIFICATE 0x00000001UL
- #define CKO\_PUBLIC\_KEY 0x00000002UL
- #define CKO\_PRIVATE\_KEY 0x00000003UL
- #define CKO\_SECRET\_KEY 0x00000004UL
- #define CKO\_HW\_FEATURE 0x00000005UL
- #define CKO\_DOMAIN\_PARAMETERS 0x00000006UL
- #define CKO\_MECHANISM 0x00000007UL
- #define CKO\_OTP\_KEY 0x00000008UL
- #define CKO\_VENDOR\_DEFINED 0x80000000UL
- #define CKH\_MONOTONIC\_COUNTER 0x00000001UL
- #define CKH\_CLOCK 0x00000002UL
- #define CKH\_USER\_INTERFACE 0x00000003UL
- #define CKH\_VENDOR\_DEFINED 0x80000000UL
- #define CKK\_RSA 0x00000000UL
- #define CKK\_DSA 0x00000001UL
- #define CKK\_DH 0x00000002UL
- #define CKK\_ECDSA 0x00000003UL /\* Deprecated \*/
- #define CKK\_EC 0x00000003UL
- #define CKK\_X9\_42\_DH 0x00000004UL
- #define CKK\_KEA 0x00000005UL

- #define [CKK\\_GENERIC\\_SECRET](#) 0x00000010UL
- #define [CKK\\_RC2](#) 0x00000011UL
- #define [CKK\\_RC4](#) 0x00000012UL
- #define [CKK\\_DES](#) 0x00000013UL
- #define [CKK\\_DES2](#) 0x00000014UL
- #define [CKK\\_DES3](#) 0x00000015UL
- #define [CKK\\_CAST](#) 0x00000016UL
- #define [CKK\\_CAST3](#) 0x00000017UL
- #define [CKK\\_CAST5](#) 0x00000018UL /\* Deprecated \*/
- #define [CKK\\_CAST128](#) 0x00000018UL
- #define [CKK\\_RC5](#) 0x00000019UL
- #define [CKK\\_IDEA](#) 0x0000001AUL
- #define [CKK\\_SKIPJACK](#) 0x0000001BUL
- #define [CKK\\_BATON](#) 0x0000001CUL
- #define [CKK\\_JUNIPER](#) 0x0000001DUL
- #define [CKK\\_CDMF](#) 0x0000001EUL
- #define [CKK\\_AES](#) 0x0000001FUL
- #define [CKK\\_BLOWFISH](#) 0x00000020UL
- #define [CKK\\_TWOFISH](#) 0x00000021UL
- #define [CKK\\_SECURID](#) 0x00000022UL
- #define [CKK\\_HOTP](#) 0x00000023UL
- #define [CKK\\_ACTI](#) 0x00000024UL
- #define [CKK\\_CAMELLIA](#) 0x00000025UL
- #define [CKK\\_ARIA](#) 0x00000026UL
- #define [CKK\\_MD5\\_HMAC](#) 0x00000027UL
- #define [CKK\\_SHA\\_1\\_HMAC](#) 0x00000028UL
- #define [CKK\\_RIPEMD128\\_HMAC](#) 0x00000029UL
- #define [CKK\\_RIPEMD160\\_HMAC](#) 0x0000002AUL
- #define [CKK\\_SHA256\\_HMAC](#) 0x0000002BUL
- #define [CKK\\_SHA384\\_HMAC](#) 0x0000002CUL
- #define [CKK\\_SHA512\\_HMAC](#) 0x0000002DUL
- #define [CKK\\_SHA224\\_HMAC](#) 0x0000002EUL
- #define [CKK\\_SEED](#) 0x0000002FUL
- #define [CKK\\_GOSTR3410](#) 0x00000030UL
- #define [CKK\\_GOSTR3411](#) 0x00000031UL
- #define [CKK\\_GOST28147](#) 0x00000032UL
- #define [CKK\\_VENDOR\\_DEFINED](#) 0x80000000UL
- #define [CK\\_CERTIFICATE\\_CATEGORY\\_UNSPECIFIED](#) 0UL
- #define [CK\\_CERTIFICATE\\_CATEGORY\\_TOKEN\\_USER](#) 1UL
- #define [CK\\_CERTIFICATE\\_CATEGORY\\_AUTHORITY](#) 2UL
- #define [CK\\_CERTIFICATE\\_CATEGORY\\_OTHER\\_ENTITY](#) 3UL
- #define [CK\\_SECURITY\\_DOMAIN\\_UNSPECIFIED](#) 0UL
- #define [CK\\_SECURITY\\_DOMAIN\\_MANUFACTURER](#) 1UL
- #define [CK\\_SECURITY\\_DOMAIN\\_OPERATOR](#) 2UL
- #define [CK\\_SECURITY\\_DOMAIN\\_THIRD\\_PARTY](#) 3UL
- #define [CKC\\_X\\_509](#) 0x00000000UL
- #define [CKC\\_X\\_509\\_ATTR\\_CERT](#) 0x00000001UL
- #define [CKC\\_WTLS](#) 0x00000002UL
- #define [CKC\\_VENDOR\\_DEFINED](#) 0x80000000UL
- #define [CKC\\_OPENPGP](#) ([CKC\\_VENDOR\\_DEFINED](#) | 0x00504750)
- #define [CKF\\_ARRAY\\_ATTRIBUTE](#) 0x40000000UL
- #define [CK\\_OTP\\_FORMAT\\_DECIMAL](#) 0UL
- #define [CK\\_OTP\\_FORMAT\\_HEXADecimal](#) 1UL
- #define [CK\\_OTP\\_FORMAT\\_ALPHANUMERIC](#) 2UL
- #define [CK\\_OTP\\_FORMAT\\_BINARY](#) 3UL



- #define CK\_OTP\_PARAM\_IGNORED 0UL
- #define CK\_OTP\_PARAM\_OPTIONAL 1UL
- #define CK\_OTP\_PARAM\_MANDATORY 2UL
- #define CKA\_CLASS 0x00000000UL
- #define CKA\_TOKEN 0x00000001UL
- #define CKA\_PRIVATE 0x00000002UL
- #define CKA\_LABEL 0x00000003UL
- #define CKA\_APPLICATION 0x00000010UL
- #define CKA\_VALUE 0x00000011UL
- #define CKA\_OBJECT\_ID 0x00000012UL
- #define CKA\_CERTIFICATE\_TYPE 0x00000080UL
- #define CKA\_ISSUER 0x00000081UL
- #define CKA\_SERIAL\_NUMBER 0x00000082UL
- #define CKA\_AC\_ISSUER 0x00000083UL
- #define CKA\_OWNER 0x00000084UL
- #define CKA\_ATTR\_TYPES 0x00000085UL
- #define CKA\_TRUSTED 0x00000086UL
- #define CKA\_CERTIFICATE\_CATEGORY 0x00000087UL
- #define CKA\_JAVA\_MIDP\_SECURITY\_DOMAIN 0x00000088UL
- #define CKA\_URL 0x00000089UL
- #define CKA\_HASH\_OF\_SUBJECT\_PUBLIC\_KEY 0x0000008AUL
- #define CKA\_HASH\_OF\_ISSUER\_PUBLIC\_KEY 0x0000008BUL
- #define CKA\_NAME\_HASH\_ALGORITHM 0x0000008CUL
- #define CKA\_CHECK\_VALUE 0x00000090UL
- #define CKA\_KEY\_TYPE 0x00000100UL
- #define CKA\_SUBJECT 0x00000101UL
- #define CKA\_ID 0x00000102UL
- #define CKA\_SENSITIVE 0x00000103UL
- #define CKA\_ENCRYPT 0x00000104UL
- #define CKA\_DECRYPT 0x00000105UL
- #define CKA\_WRAP 0x00000106UL
- #define CKA\_UNWRAP 0x00000107UL
- #define CKA\_SIGN 0x00000108UL
- #define CKA\_SIGN\_RECOVER 0x00000109UL
- #define CKA\_VERIFY 0x0000010AUL
- #define CKA\_VERIFY\_RECOVER 0x0000010BUL
- #define CKA\_DERIVE 0x0000010CUL
- #define CKA\_START\_DATE 0x00000110UL
- #define CKA\_END\_DATE 0x00000111UL
- #define CKA\_MODULUS 0x00000120UL
- #define CKA\_MODULUS\_BITS 0x00000121UL
- #define CKA\_PUBLIC\_EXPONENT 0x00000122UL
- #define CKA\_PRIVATE\_EXPONENT 0x00000123UL
- #define CKA\_PRIME\_1 0x00000124UL
- #define CKA\_PRIME\_2 0x00000125UL
- #define CKA\_EXPONENT\_1 0x00000126UL
- #define CKA\_EXPONENT\_2 0x00000127UL
- #define CKA\_COEFFICIENT 0x00000128UL
- #define CKA\_PUBLIC\_KEY\_INFO 0x00000129UL
- #define CKA\_PRIME 0x00000130UL
- #define CKA\_SUBPRIME 0x00000131UL
- #define CKA\_BASE 0x00000132UL
- #define CKA\_PRIME\_BITS 0x00000133UL
- #define CKA\_SUBPRIME\_BITS 0x00000134UL
- #define CKA\_SUB\_PRIME\_BITS CKA\_SUBPRIME\_BITS

- #define CKA\_VALUE\_BITS 0x00000160UL
- #define CKA\_VALUE\_LEN 0x00000161UL
- #define CKA\_EXTRACTABLE 0x00000162UL
- #define CKA\_LOCAL 0x00000163UL
- #define CKA\_NEVER\_EXTRACTABLE 0x00000164UL
- #define CKA\_ALWAYS\_SENSITIVE 0x00000165UL
- #define CKA\_KEY\_GEN\_MECHANISM 0x00000166UL
- #define CKA\_MODIFIABLE 0x00000170UL
- #define CKA\_COPYABLE 0x00000171UL
- #define CKA\_DESTROYABLE 0x00000172UL
- #define CKA\_ECDSA\_PARAMS 0x00000180UL /\* Deprecated \*/
- #define CKA\_EC\_PARAMS 0x00000180UL
- #define CKA\_EC\_POINT 0x00000181UL
- #define CKA\_SECONDARY\_AUTH 0x00000200UL /\* Deprecated \*/
- #define CKA\_AUTH\_PIN\_FLAGS 0x00000201UL /\* Deprecated \*/
- #define CKA\_ALWAYS\_AUTHENTICATE 0x00000202UL
- #define CKA\_WRAP\_WITH\_TRUSTED 0x00000210UL
- #define CKA\_WRAP\_TEMPLATE (CKF\_ARRAY\_ATTRIBUTE | 0x00000211UL)
- #define CKA\_UNWRAP\_TEMPLATE (CKF\_ARRAY\_ATTRIBUTE | 0x00000212UL)
- #define CKA\_DERIVE\_TEMPLATE (CKF\_ARRAY\_ATTRIBUTE | 0x00000213UL)
- #define CKA\_OTP\_FORMAT 0x00000220UL
- #define CKA\_OTP\_LENGTH 0x00000221UL
- #define CKA\_OTP\_TIME\_INTERVAL 0x00000222UL
- #define CKA\_OTP\_USER\_FRIENDLY\_MODE 0x00000223UL
- #define CKA\_OTP\_CHALLENGE\_REQUIREMENT 0x00000224UL
- #define CKA\_OTP\_TIME\_REQUIREMENT 0x00000225UL
- #define CKA\_OTP\_COUNTER\_REQUIREMENT 0x00000226UL
- #define CKA\_OTP\_PIN\_REQUIREMENT 0x00000227UL
- #define CKA\_OTP\_COUNTER 0x0000022EUL
- #define CKA\_OTP\_TIME 0x0000022FUL
- #define CKA\_OTP\_USER\_IDENTIFIER 0x0000022AUL
- #define CKA\_OTP\_SERVICE\_IDENTIFIER 0x0000022BUL
- #define CKA\_OTP\_SERVICE\_LOGO 0x0000022CUL
- #define CKA\_OTP\_SERVICE\_LOGO\_TYPE 0x0000022DUL
- #define CKA\_GOSTR3410\_PARAMS 0x00000250UL
- #define CKA\_GOSTR3411\_PARAMS 0x00000251UL
- #define CKA\_GOST28147\_PARAMS 0x00000252UL
- #define CKA\_HW\_FEATURE\_TYPE 0x00000300UL
- #define CKA\_RESET\_ON\_INIT 0x00000301UL
- #define CKA\_HAS\_RESET 0x00000302UL
- #define CKA\_PIXEL\_X 0x00000400UL
- #define CKA\_PIXEL\_Y 0x00000401UL
- #define CKA\_RESOLUTION 0x00000402UL
- #define CKA\_CHAR\_ROWS 0x00000403UL
- #define CKA\_CHAR\_COLUMNS 0x00000404UL
- #define CKA\_COLOR 0x00000405UL
- #define CKA\_BITS\_PER\_PIXEL 0x00000406UL
- #define CKA\_CHAR\_SETS 0x00000480UL
- #define CKA\_ENCODING\_METHODS 0x00000481UL
- #define CKA\_MIME\_TYPES 0x00000482UL
- #define CKA\_MECHANISM\_TYPE 0x00000500UL
- #define CKA\_REQUIRED\_CMS\_ATTRIBUTES 0x00000501UL
- #define CKA\_DEFAULT\_CMS\_ATTRIBUTES 0x00000502UL
- #define CKA\_SUPPORTED\_CMS\_ATTRIBUTES 0x00000503UL
- #define CKA\_ALLOWED\_MECHANISMS (CKF\_ARRAY\_ATTRIBUTE | 0x00000600UL)

- #define CKA\_VENDOR\_DEFINED 0x80000000UL
- #define CKM\_RSA\_PKCS\_KEY\_PAIR\_GEN 0x00000000UL
- #define CKM\_RSA\_PKCS 0x00000001UL
- #define CKM\_RSA\_9796 0x00000002UL
- #define CKM\_RSA\_X\_509 0x00000003UL
- #define CKM\_MD2\_RSA\_PKCS 0x00000004UL
- #define CKM\_MD5\_RSA\_PKCS 0x00000005UL
- #define CKM\_SHA1\_RSA\_PKCS 0x00000006UL
- #define CKM\_RIPMD128\_RSA\_PKCS 0x00000007UL
- #define CKM\_RIPMD160\_RSA\_PKCS 0x00000008UL
- #define CKM\_RSA\_PKCS\_OAEP 0x00000009UL
- #define CKM\_RSA\_X9\_31\_KEY\_PAIR\_GEN 0x0000000AUL
- #define CKM\_RSA\_X9\_31 0x0000000BUL
- #define CKM\_SHA1\_RSA\_X9\_31 0x0000000CUL
- #define CKM\_RSA\_PKCS\_PSS 0x0000000DUL
- #define CKM\_SHA1\_RSA\_PKCS\_PSS 0x0000000EUL
- #define CKM\_DSA\_KEY\_PAIR\_GEN 0x00000010UL
- #define CKM\_DSA 0x00000011UL
- #define CKM\_DSA\_SHA1 0x00000012UL
- #define CKM\_DSA\_SHA224 0x00000013UL
- #define CKM\_DSA\_SHA256 0x00000014UL
- #define CKM\_DSA\_SHA384 0x00000015UL
- #define CKM\_DSA\_SHA512 0x00000016UL
- #define CKM\_DH\_PKCS\_KEY\_PAIR\_GEN 0x00000020UL
- #define CKM\_DH\_PKCS\_DERIVE 0x00000021UL
- #define CKM\_X9\_42\_DH\_KEY\_PAIR\_GEN 0x00000030UL
- #define CKM\_X9\_42\_DH\_DERIVE 0x00000031UL
- #define CKM\_X9\_42\_DH\_HYBRID\_DERIVE 0x00000032UL
- #define CKM\_X9\_42\_MQV\_DERIVE 0x00000033UL
- #define CKM\_SHA256\_RSA\_PKCS 0x00000040UL
- #define CKM\_SHA384\_RSA\_PKCS 0x00000041UL
- #define CKM\_SHA512\_RSA\_PKCS 0x00000042UL
- #define CKM\_SHA256\_RSA\_PKCS\_PSS 0x00000043UL
- #define CKM\_SHA384\_RSA\_PKCS\_PSS 0x00000044UL
- #define CKM\_SHA512\_RSA\_PKCS\_PSS 0x00000045UL
- #define CKM\_SHA224\_RSA\_PKCS 0x00000046UL
- #define CKM\_SHA224\_RSA\_PKCS\_PSS 0x00000047UL
- #define CKM\_SHA512\_224 0x00000048UL
- #define CKM\_SHA512\_224\_HMAC 0x00000049UL
- #define CKM\_SHA512\_224\_HMAC\_GENERAL 0x0000004AUL
- #define CKM\_SHA512\_224\_KEY\_DERIVATION 0x0000004BUL
- #define CKM\_SHA512\_256 0x0000004CUL
- #define CKM\_SHA512\_256\_HMAC 0x0000004DUL
- #define CKM\_SHA512\_256\_HMAC\_GENERAL 0x0000004EUL
- #define CKM\_SHA512\_256\_KEY\_DERIVATION 0x0000004FUL
- #define CKM\_SHA512\_T 0x00000050UL
- #define CKM\_SHA512\_T\_HMAC 0x00000051UL
- #define CKM\_SHA512\_T\_HMAC\_GENERAL 0x00000052UL
- #define CKM\_SHA512\_T\_KEY\_DERIVATION 0x00000053UL
- #define CKM\_RC2\_KEY\_GEN 0x00000100UL
- #define CKM\_RC2\_ECB 0x00000101UL
- #define CKM\_RC2\_CBC 0x00000102UL
- #define CKM\_RC2\_MAC 0x00000103UL
- #define CKM\_RC2\_MAC\_GENERAL 0x00000104UL
- #define CKM\_RC2\_CBC\_PAD 0x00000105UL

- #define CKM\_RC4\_KEY\_GEN 0x00000110UL
- #define CKM\_RC4 0x00000111UL
- #define CKM\_DES\_KEY\_GEN 0x00000120UL
- #define CKM\_DES\_ECB 0x00000121UL
- #define CKM\_DES\_CBC 0x00000122UL
- #define CKM\_DES\_MAC 0x00000123UL
- #define CKM\_DES\_MAC\_GENERAL 0x00000124UL
- #define CKM\_DES\_CBC\_PAD 0x00000125UL
- #define CKM\_DES2\_KEY\_GEN 0x00000130UL
- #define CKM\_DES3\_KEY\_GEN 0x00000131UL
- #define CKM\_DES3\_ECB 0x00000132UL
- #define CKM\_DES3\_CBC 0x00000133UL
- #define CKM\_DES3\_MAC 0x00000134UL
- #define CKM\_DES3\_MAC\_GENERAL 0x00000135UL
- #define CKM\_DES3\_CBC\_PAD 0x00000136UL
- #define CKM\_DES3\_CMAC\_GENERAL 0x00000137UL
- #define CKM\_DES3\_CMAC 0x00000138UL
- #define CKM\_CDMF\_KEY\_GEN 0x00000140UL
- #define CKM\_CDMF\_ECB 0x00000141UL
- #define CKM\_CDMF\_CBC 0x00000142UL
- #define CKM\_CDMF\_MAC 0x00000143UL
- #define CKM\_CDMF\_MAC\_GENERAL 0x00000144UL
- #define CKM\_CDMF\_CBC\_PAD 0x00000145UL
- #define CKM\_DES\_OFB64 0x00000150UL
- #define CKM\_DES\_OFB8 0x00000151UL
- #define CKM\_DES\_CFB64 0x00000152UL
- #define CKM\_DES\_CFB8 0x00000153UL
- #define CKM\_MD2 0x00000200UL
- #define CKM\_MD2\_HMAC 0x00000201UL
- #define CKM\_MD2\_HMAC\_GENERAL 0x00000202UL
- #define CKM\_MD5 0x00000210UL
- #define CKM\_MD5\_HMAC 0x00000211UL
- #define CKM\_MD5\_HMAC\_GENERAL 0x00000212UL
- #define CKM\_SHA\_1 0x00000220UL
- #define CKM\_SHA\_1\_HMAC 0x00000221UL
- #define CKM\_SHA\_1\_HMAC\_GENERAL 0x00000222UL
- #define CKM\_RIPEMD128 0x00000230UL
- #define CKM\_RIPEMD128\_HMAC 0x00000231UL
- #define CKM\_RIPEMD128\_HMAC\_GENERAL 0x00000232UL
- #define CKM\_RIPEMD160 0x00000240UL
- #define CKM\_RIPEMD160\_HMAC 0x00000241UL
- #define CKM\_RIPEMD160\_HMAC\_GENERAL 0x00000242UL
- #define CKM\_SHA256 0x00000250UL
- #define CKM\_SHA256\_HMAC 0x00000251UL
- #define CKM\_SHA256\_HMAC\_GENERAL 0x00000252UL
- #define CKM\_SHA224 0x00000255UL
- #define CKM\_SHA224\_HMAC 0x00000256UL
- #define CKM\_SHA224\_HMAC\_GENERAL 0x00000257UL
- #define CKM\_SHA384 0x00000260UL
- #define CKM\_SHA384\_HMAC 0x00000261UL
- #define CKM\_SHA384\_HMAC\_GENERAL 0x00000262UL
- #define CKM\_SHA512 0x00000270UL
- #define CKM\_SHA512\_HMAC 0x00000271UL
- #define CKM\_SHA512\_HMAC\_GENERAL 0x00000272UL
- #define CKM\_SECURID\_KEY\_GEN 0x00000280UL

- #define CKM\_SECURID 0x00000282UL
- #define CKM\_HOTP\_KEY\_GEN 0x00000290UL
- #define CKM\_HOTP 0x00000291UL
- #define CKM\_ACTI 0x000002A0UL
- #define CKM\_ACTI\_KEY\_GEN 0x000002A1UL
- #define CKM\_CAST\_KEY\_GEN 0x00000300UL
- #define CKM\_CAST\_ECB 0x00000301UL
- #define CKM\_CAST\_CBC 0x00000302UL
- #define CKM\_CAST\_MAC 0x00000303UL
- #define CKM\_CAST\_MAC\_GENERAL 0x00000304UL
- #define CKM\_CAST\_CBC\_PAD 0x00000305UL
- #define CKM\_CAST3\_KEY\_GEN 0x00000310UL
- #define CKM\_CAST3\_ECB 0x00000311UL
- #define CKM\_CAST3\_CBC 0x00000312UL
- #define CKM\_CAST3\_MAC 0x00000313UL
- #define CKM\_CAST3\_MAC\_GENERAL 0x00000314UL
- #define CKM\_CAST3\_CBC\_PAD 0x00000315UL
- #define CKM\_CAST5\_KEY\_GEN 0x00000320UL
- #define CKM\_CAST128\_KEY\_GEN 0x00000320UL
- #define CKM\_CAST5\_ECB 0x00000321UL
- #define CKM\_CAST128\_ECB 0x00000321UL
- #define CKM\_CAST5\_CBC 0x00000322UL /\* Deprecated \*/
- #define CKM\_CAST128\_CBC 0x00000322UL
- #define CKM\_CAST5\_MAC 0x00000323UL /\* Deprecated \*/
- #define CKM\_CAST128\_MAC 0x00000323UL
- #define CKM\_CAST5\_MAC\_GENERAL 0x00000324UL /\* Deprecated \*/
- #define CKM\_CAST128\_MAC\_GENERAL 0x00000324UL
- #define CKM\_CAST5\_CBC\_PAD 0x00000325UL /\* Deprecated \*/
- #define CKM\_CAST128\_CBC\_PAD 0x00000325UL
- #define CKM\_RC5\_KEY\_GEN 0x00000330UL
- #define CKM\_RC5\_ECB 0x00000331UL
- #define CKM\_RC5\_CBC 0x00000332UL
- #define CKM\_RC5\_MAC 0x00000333UL
- #define CKM\_RC5\_MAC\_GENERAL 0x00000334UL
- #define CKM\_RC5\_CBC\_PAD 0x00000335UL
- #define CKM\_IDEA\_KEY\_GEN 0x00000340UL
- #define CKM\_IDEA\_ECB 0x00000341UL
- #define CKM\_IDEA\_CBC 0x00000342UL
- #define CKM\_IDEA\_MAC 0x00000343UL
- #define CKM\_IDEA\_MAC\_GENERAL 0x00000344UL
- #define CKM\_IDEA\_CBC\_PAD 0x00000345UL
- #define CKM\_GENERIC\_SECRET\_KEY\_GEN 0x00000350UL
- #define CKM\_CONCATENATE\_BASE\_AND\_KEY 0x00000360UL
- #define CKM\_CONCATENATE\_BASE\_AND\_DATA 0x00000362UL
- #define CKM\_CONCATENATE\_DATA\_AND\_BASE 0x00000363UL
- #define CKM\_XOR\_BASE\_AND\_DATA 0x00000364UL
- #define CKM\_EXTRACT\_KEY\_FROM\_KEY 0x00000365UL
- #define CKM\_SSL3\_PRE\_MASTER\_KEY\_GEN 0x00000370UL
- #define CKM\_SSL3\_MASTER\_KEY\_DERIVE 0x00000371UL
- #define CKM\_SSL3\_KEY\_AND\_MAC\_DERIVE 0x00000372UL
- #define CKM\_SSL3\_MASTER\_KEY\_DERIVE\_DH 0x00000373UL
- #define CKM\_TLS\_PRE\_MASTER\_KEY\_GEN 0x00000374UL
- #define CKM\_TLS\_MASTER\_KEY\_DERIVE 0x00000375UL
- #define CKM\_TLS\_KEY\_AND\_MAC\_DERIVE 0x00000376UL
- #define CKM\_TLS\_MASTER\_KEY\_DERIVE\_DH 0x00000377UL

- #define CKM\_TLS\_PRF 0x00000378UL
- #define CKM\_SSL3\_MD5\_MAC 0x00000380UL
- #define CKM\_SSL3\_SHA1\_MAC 0x00000381UL
- #define CKM\_MD5\_KEY\_DERIVATION 0x00000390UL
- #define CKM\_MD2\_KEY\_DERIVATION 0x00000391UL
- #define CKM\_SHA1\_KEY\_DERIVATION 0x00000392UL
- #define CKM\_SHA256\_KEY\_DERIVATION 0x00000393UL
- #define CKM\_SHA384\_KEY\_DERIVATION 0x00000394UL
- #define CKM\_SHA512\_KEY\_DERIVATION 0x00000395UL
- #define CKM\_SHA224\_KEY\_DERIVATION 0x00000396UL
- #define CKM\_PBE\_MD2\_DES\_CBC 0x000003A0UL
- #define CKM\_PBE\_MD5\_DES\_CBC 0x000003A1UL
- #define CKM\_PBE\_MD5\_CAST\_CBC 0x000003A2UL
- #define CKM\_PBE\_MD5\_CAST3\_CBC 0x000003A3UL
- #define CKM\_PBE\_MD5\_CAST5\_CBC 0x000003A4UL /\* Deprecated \*/
- #define CKM\_PBE\_MD5\_CAST128\_CBC 0x000003A4UL
- #define CKM\_PBE\_SHA1\_CAST5\_CBC 0x000003A5UL /\* Deprecated \*/
- #define CKM\_PBE\_SHA1\_CAST128\_CBC 0x000003A5UL
- #define CKM\_PBE\_SHA1\_RC4\_128 0x000003A6UL
- #define CKM\_PBE\_SHA1\_RC4\_40 0x000003A7UL
- #define CKM\_PBE\_SHA1\_DES3\_EDE\_CBC 0x000003A8UL
- #define CKM\_PBE\_SHA1\_DES2\_EDE\_CBC 0x000003A9UL
- #define CKM\_PBE\_SHA1\_RC2\_128\_CBC 0x000003AAUL
- #define CKM\_PBE\_SHA1\_RC2\_40\_CBC 0x000003ABUL
- #define CKM\_PKCS5\_PBKD2 0x000003B0UL
- #define CKM\_PBA\_SHA1\_WITH\_SHA1\_HMAC 0x000003C0UL
- #define CKM\_WTLS\_PRE\_MASTER\_KEY\_GEN 0x000003D0UL
- #define CKM\_WTLS\_MASTER\_KEY\_DERIVE 0x000003D1UL
- #define CKM\_WTLS\_MASTER\_KEY\_DERIVE\_DH\_ECC 0x000003D2UL
- #define CKM\_WTLS\_PRF 0x000003D3UL
- #define CKM\_WTLS\_SERVER\_KEY\_AND\_MAC\_DERIVE 0x000003D4UL
- #define CKM\_WTLS\_CLIENT\_KEY\_AND\_MAC\_DERIVE 0x000003D5UL
- #define CKM\_TLS10\_MAC\_SERVER 0x000003D6UL
- #define CKM\_TLS10\_MAC\_CLIENT 0x000003D7UL
- #define CKM\_TLS12\_MAC 0x000003D8UL
- #define CKM\_TLS12\_KDF 0x000003D9UL
- #define CKM\_TLS12\_MASTER\_KEY\_DERIVE 0x000003E0UL
- #define CKM\_TLS12\_KEY\_AND\_MAC\_DERIVE 0x000003E1UL
- #define CKM\_TLS12\_MASTER\_KEY\_DERIVE\_DH 0x000003E2UL
- #define CKM\_TLS12\_KEY\_SAFE\_DERIVE 0x000003E3UL
- #define CKM\_TLS\_MAC 0x000003E4UL
- #define CKM\_TLS\_KDF 0x000003E5UL
- #define CKM\_KEY\_WRAP\_LYNKS 0x00000400UL
- #define CKM\_KEY\_WRAP\_SET\_OAEP 0x00000401UL
- #define CKM\_CMS\_SIG 0x00000500UL
- #define CKM\_KIP\_DERIVE 0x00000510UL
- #define CKM\_KIP\_WRAP 0x00000511UL
- #define CKM\_KIP\_MAC 0x00000512UL
- #define CKM\_CAMELLIA\_KEY\_GEN 0x00000550UL
- #define CKM\_CAMELLIA\_ECB 0x00000551UL
- #define CKM\_CAMELLIA\_CBC 0x00000552UL
- #define CKM\_CAMELLIA\_MAC 0x00000553UL
- #define CKM\_CAMELLIA\_MAC\_GENERAL 0x00000554UL
- #define CKM\_CAMELLIA\_CBC\_PAD 0x00000555UL
- #define CKM\_CAMELLIA\_ECB\_ENCRYPT\_DATA 0x00000556UL



- #define CKM\_CAMELLIA\_CBC\_ENCRYPT\_DATA 0x00000557UL
- #define CKM\_CAMELLIA\_CTR 0x00000558UL
- #define CKM\_ARIA\_KEY\_GEN 0x00000560UL
- #define CKM\_ARIA\_ECB 0x00000561UL
- #define CKM\_ARIA\_CBC 0x00000562UL
- #define CKM\_ARIA\_MAC 0x00000563UL
- #define CKM\_ARIA\_MAC\_GENERAL 0x00000564UL
- #define CKM\_ARIA\_CBC\_PAD 0x00000565UL
- #define CKM\_ARIA\_ECB\_ENCRYPT\_DATA 0x00000566UL
- #define CKM\_ARIA\_CBC\_ENCRYPT\_DATA 0x00000567UL
- #define CKM\_SEED\_KEY\_GEN 0x00000650UL
- #define CKM\_SEED\_ECB 0x00000651UL
- #define CKM\_SEED\_CBC 0x00000652UL
- #define CKM\_SEED\_MAC 0x00000653UL
- #define CKM\_SEED\_MAC\_GENERAL 0x00000654UL
- #define CKM\_SEED\_CBC\_PAD 0x00000655UL
- #define CKM\_SEED\_ECB\_ENCRYPT\_DATA 0x00000656UL
- #define CKM\_SEED\_CBC\_ENCRYPT\_DATA 0x00000657UL
- #define CKM\_SKIPJACK\_KEY\_GEN 0x00001000UL
- #define CKM\_SKIPJACK\_ECB64 0x00001001UL
- #define CKM\_SKIPJACK\_CBC64 0x00001002UL
- #define CKM\_SKIPJACK\_OFB64 0x00001003UL
- #define CKM\_SKIPJACK\_CFB64 0x00001004UL
- #define CKM\_SKIPJACK\_CFB32 0x00001005UL
- #define CKM\_SKIPJACK\_CFB16 0x00001006UL
- #define CKM\_SKIPJACK\_CFB8 0x00001007UL
- #define CKM\_SKIPJACK\_WRAP 0x00001008UL
- #define CKM\_SKIPJACK\_PRIVATE\_WRAP 0x00001009UL
- #define CKM\_SKIPJACK\_RELAYX 0x0000100aUL
- #define CKM\_KEA\_KEY\_PAIR\_GEN 0x00001010UL
- #define CKM\_KEA\_KEY\_DERIVE 0x00001011UL
- #define CKM\_KEA\_DERIVE 0x00001012UL
- #define CKM\_FORTEZZA\_TIMESTAMP 0x00001020UL
- #define CKM\_BATON\_KEY\_GEN 0x00001030UL
- #define CKM\_BATON\_ECB128 0x00001031UL
- #define CKM\_BATON\_ECB96 0x00001032UL
- #define CKM\_BATON\_CBC128 0x00001033UL
- #define CKM\_BATON\_COUNTER 0x00001034UL
- #define CKM\_BATON\_SHUFFLE 0x00001035UL
- #define CKM\_BATON\_WRAP 0x00001036UL
- #define CKM\_ECDSA\_KEY\_PAIR\_GEN 0x00001040UL /\* Deprecated \*/
- #define CKM\_EC\_KEY\_PAIR\_GEN 0x00001040UL
- #define CKM\_ECDSA 0x00001041UL
- #define CKM\_ECDSA\_SHA1 0x00001042UL
- #define CKM\_ECDSA\_SHA224 0x00001043UL
- #define CKM\_ECDSA\_SHA256 0x00001044UL
- #define CKM\_ECDSA\_SHA384 0x00001045UL
- #define CKM\_ECDSA\_SHA512 0x00001046UL
- #define CKM\_ECDH1\_DERIVE 0x00001050UL
- #define CKM\_ECDH1\_COFACTOR\_DERIVE 0x00001051UL
- #define CKM\_ECMQV\_DERIVE 0x00001052UL
- #define CKM\_ECDH\_AES\_KEY\_WRAP 0x00001053UL
- #define CKM\_RSA\_AES\_KEY\_WRAP 0x00001054UL
- #define CKM\_JUNIPER\_KEY\_GEN 0x00001060UL
- #define CKM\_JUNIPER\_ECB128 0x00001061UL

- #define CKM\_JUNIPER\_CBC128 0x00001062UL
- #define CKM\_JUNIPER\_COUNTER 0x00001063UL
- #define CKM\_JUNIPER\_SHUFFLE 0x00001064UL
- #define CKM\_JUNIPER\_WRAP 0x00001065UL
- #define CKM\_FASTHASH 0x00001070UL
- #define CKM\_AES\_KEY\_GEN 0x00001080UL
- #define CKM\_AES\_ECB 0x00001081UL
- #define CKM\_AES\_CBC 0x00001082UL
- #define CKM\_AES\_MAC 0x00001083UL
- #define CKM\_AES\_MAC\_GENERAL 0x00001084UL
- #define CKM\_AES\_CBC\_PAD 0x00001085UL
- #define CKM\_AES\_CTR 0x00001086UL
- #define CKM\_AES\_GCM 0x00001087UL
- #define CKM\_AES\_CCM 0x00001088UL
- #define CKM\_AES\_CTS 0x00001089UL
- #define CKM\_AES\_CMAC 0x0000108AUL
- #define CKM\_AES\_CMAC\_GENERAL 0x0000108BUL
- #define CKM\_AES\_XCBC\_MAC 0x0000108CUL
- #define CKM\_AES\_XCBC\_MAC\_96 0x0000108DUL
- #define CKM\_AES\_GMAC 0x0000108EUL
- #define CKM\_BLOWFISH\_KEY\_GEN 0x00001090UL
- #define CKM\_BLOWFISH\_CBC 0x00001091UL
- #define CKM\_TWOFISH\_KEY\_GEN 0x00001092UL
- #define CKM\_TWOFISH\_CBC 0x00001093UL
- #define CKM\_BLOWFISH\_CBC\_PAD 0x00001094UL
- #define CKM\_TWOFISH\_CBC\_PAD 0x00001095UL
- #define CKM\_DES\_ECB\_ENCRYPT\_DATA 0x00001100UL
- #define CKM\_DES\_CBC\_ENCRYPT\_DATA 0x00001101UL
- #define CKM\_DES3\_ECB\_ENCRYPT\_DATA 0x00001102UL
- #define CKM\_DES3\_CBC\_ENCRYPT\_DATA 0x00001103UL
- #define CKM\_AES\_ECB\_ENCRYPT\_DATA 0x00001104UL
- #define CKM\_AES\_CBC\_ENCRYPT\_DATA 0x00001105UL
- #define CKM\_GOSTR3410\_KEY\_PAIR\_GEN 0x00001200UL
- #define CKM\_GOSTR3410 0x00001201UL
- #define CKM\_GOSTR3410\_WITH\_GOSTR3411 0x00001202UL
- #define CKM\_GOSTR3410\_KEY\_WRAP 0x00001203UL
- #define CKM\_GOSTR3410\_DERIVE 0x00001204UL
- #define CKM\_GOSTR3411 0x00001210UL
- #define CKM\_GOSTR3411\_HMAC 0x00001211UL
- #define CKM\_GOST28147\_KEY\_GEN 0x00001220UL
- #define CKM\_GOST28147\_ECB 0x00001221UL
- #define CKM\_GOST28147 0x00001222UL
- #define CKM\_GOST28147\_MAC 0x00001223UL
- #define CKM\_GOST28147\_KEY\_WRAP 0x00001224UL
- #define CKM\_DSA\_PARAMETER\_GEN 0x00002000UL
- #define CKM\_DH\_PKCS\_PARAMETER\_GEN 0x00002001UL
- #define CKM\_X9\_42\_DH\_PARAMETER\_GEN 0x00002002UL
- #define CKM\_DSA\_PROBABLISTIC\_PARAMETER\_GEN 0x00002003UL
- #define CKM\_DSA\_SHAW\_TAYLOR\_PARAMETER\_GEN 0x00002004UL
- #define CKM\_AES\_OFB 0x00002104UL
- #define CKM\_AES\_CFB64 0x00002105UL
- #define CKM\_AES\_CFB8 0x00002106UL
- #define CKM\_AES\_CFB128 0x00002107UL
- #define CKM\_AES\_CFB1 0x00002108UL
- #define CKM\_AES\_KEY\_WRAP 0x00002109UL /\* WAS: 0x00001090 \*/



- #define CKM\_AES\_KEY\_WRAP\_PAD 0x0000210AUL /\* WAS: 0x00001091 \*/
- #define CKM\_RSA\_PKCS\_TPM\_1\_1 0x00004001UL
- #define CKM\_RSA\_PKCS\_OAEP\_TPM\_1\_1 0x00004002UL
- #define CKM\_VENDOR\_DEFINED 0x80000000UL
- #define CKF\_HW 0x00000001UL /\* performed by HW \*/
- #define CKF\_ENCRYPT 0x00000100UL
- #define CKF\_DECRYPT 0x00000200UL
- #define CKF\_DIGEST 0x00000400UL
- #define CKF\_SIGN 0x00000800UL
- #define CKF\_SIGN\_RECOVER 0x00001000UL
- #define CKF\_VERIFY 0x00002000UL
- #define CKF\_VERIFY\_RECOVER 0x00004000UL
- #define CKF\_GENERATE 0x00008000UL
- #define CKF\_GENERATE\_KEY\_PAIR 0x00010000UL
- #define CKF\_WRAP 0x00020000UL
- #define CKF\_UNWRAP 0x00040000UL
- #define CKF\_DERIVE 0x00080000UL
- #define CKF\_EC\_F\_P 0x00100000UL
- #define CKF\_EC\_F\_2M 0x00200000UL
- #define CKF\_EC\_ECPARAMETERS 0x00400000UL
- #define CKF\_EC\_NAMEDCURVE 0x00800000UL
- #define CKF\_EC\_UNCOMPRESS 0x01000000UL
- #define CKF\_EC\_COMPRESS 0x02000000UL
- #define CKF\_EXTENSION 0x80000000UL
- #define CKR\_OK 0x00000000UL
- #define CKR\_CANCEL 0x00000001UL
- #define CKR\_HOST\_MEMORY 0x00000002UL
- #define CKR\_SLOT\_ID\_INVALID 0x00000003UL
- #define CKR\_GENERAL\_ERROR 0x00000005UL
- #define CKR\_FUNCTION\_FAILED 0x00000006UL
- #define CKR\_ARGUMENTS\_BAD 0x00000007UL
- #define CKR\_NO\_EVENT 0x00000008UL
- #define CKR\_NEED\_TO\_CREATE\_THREADS 0x00000009UL
- #define CKR\_CANT\_LOCK 0x0000000AUL
- #define CKR\_ATTRIBUTE\_READ\_ONLY 0x00000010UL
- #define CKR\_ATTRIBUTE\_SENSITIVE 0x00000011UL
- #define CKR\_ATTRIBUTE\_TYPE\_INVALID 0x00000012UL
- #define CKR\_ATTRIBUTE\_VALUE\_INVALID 0x00000013UL
- #define CKR\_ACTION\_PROHIBITED 0x0000001BUL
- #define CKR\_DATA\_INVALID 0x00000020UL
- #define CKR\_DATA\_LEN\_RANGE 0x00000021UL
- #define CKR\_DEVICE\_ERROR 0x00000030UL
- #define CKR\_DEVICE\_MEMORY 0x00000031UL
- #define CKR\_DEVICE\_REMOVED 0x00000032UL
- #define CKR\_ENCRYPTED\_DATA\_INVALID 0x00000040UL
- #define CKR\_ENCRYPTED\_DATA\_LEN\_RANGE 0x00000041UL
- #define CKR\_FUNCTION\_CANCELED 0x00000050UL
- #define CKR\_FUNCTION\_NOT\_PARALLEL 0x00000051UL
- #define CKR\_FUNCTION\_NOT\_SUPPORTED 0x00000054UL
- #define CKR\_KEY\_HANDLE\_INVALID 0x00000060UL
- #define CKR\_KEY\_SIZE\_RANGE 0x00000062UL
- #define CKR\_KEY\_TYPE\_INCONSISTENT 0x00000063UL
- #define CKR\_KEY\_NOT\_NEEDED 0x00000064UL
- #define CKR\_KEY\_CHANGED 0x00000065UL
- #define CKR\_KEY\_NEEDED 0x00000066UL

- #define CKR\_KEY\_INDIGESTIBLE 0x00000067UL
- #define CKR\_KEY\_FUNCTION\_NOT\_PERMITTED 0x00000068UL
- #define CKR\_KEY\_NOT\_WRAPPABLE 0x00000069UL
- #define CKR\_KEY\_UNEXTRACTABLE 0x0000006AUL
- #define CKR\_MECHANISM\_INVALID 0x00000070UL
- #define CKR\_MECHANISM\_PARAM\_INVALID 0x00000071UL
- #define CKR\_OBJECT\_HANDLE\_INVALID 0x00000082UL
- #define CKR\_OPERATION\_ACTIVE 0x00000090UL
- #define CKR\_OPERATION\_NOT\_INITIALIZED 0x00000091UL
- #define CKR\_PIN\_INCORRECT 0x000000A0UL
- #define CKR\_PIN\_INVALID 0x000000A1UL
- #define CKR\_PIN\_LEN\_RANGE 0x000000A2UL
- #define CKR\_PIN\_EXPIRED 0x000000A3UL
- #define CKR\_PIN\_LOCKED 0x000000A4UL
- #define CKR\_SESSION\_CLOSED 0x000000B0UL
- #define CKR\_SESSION\_COUNT 0x000000B1UL
- #define CKR\_SESSION\_HANDLE\_INVALID 0x000000B3UL
- #define CKR\_SESSION\_PARALLEL\_NOT\_SUPPORTED 0x000000B4UL
- #define CKR\_SESSION\_READ\_ONLY 0x000000B5UL
- #define CKR\_SESSION\_EXISTS 0x000000B6UL
- #define CKR\_SESSION\_READ\_ONLY\_EXISTS 0x000000B7UL
- #define CKR\_SESSION\_READ\_WRITE\_SO\_EXISTS 0x000000B8UL
- #define CKR\_SIGNATURE\_INVALID 0x000000C0UL
- #define CKR\_SIGNATURE\_LEN\_RANGE 0x000000C1UL
- #define CKR\_TEMPLATE\_INCOMPLETE 0x000000D0UL
- #define CKR\_TEMPLATE\_INCONSISTENT 0x000000D1UL
- #define CKR\_TOKEN\_NOT\_PRESENT 0x000000E0UL
- #define CKR\_TOKEN\_NOT\_RECOGNIZED 0x000000E1UL
- #define CKR\_TOKEN\_WRITE\_PROTECTED 0x000000E2UL
- #define CKR\_UNWRAPPING\_KEY\_HANDLE\_INVALID 0x000000F0UL
- #define CKR\_UNWRAPPING\_KEY\_SIZE\_RANGE 0x000000F1UL
- #define CKR\_UNWRAPPING\_KEY\_TYPE\_INCONSISTENT 0x000000F2UL
- #define CKR\_USER\_ALREADY\_LOGGED\_IN 0x00000100UL
- #define CKR\_USER\_NOT\_LOGGED\_IN 0x00000101UL
- #define CKR\_USER\_PIN\_NOT\_INITIALIZED 0x00000102UL
- #define CKR\_USER\_TYPE\_INVALID 0x00000103UL
- #define CKR\_USER\_ANOTHER\_ALREADY\_LOGGED\_IN 0x00000104UL
- #define CKR\_USER\_TOO\_MANY\_TYPES 0x00000105UL
- #define CKR\_WRAPPED\_KEY\_INVALID 0x00000110UL
- #define CKR\_WRAPPED\_KEY\_LEN\_RANGE 0x00000112UL
- #define CKR\_WRAPPING\_KEY\_HANDLE\_INVALID 0x00000113UL
- #define CKR\_WRAPPING\_KEY\_SIZE\_RANGE 0x00000114UL
- #define CKR\_WRAPPING\_KEY\_TYPE\_INCONSISTENT 0x00000115UL
- #define CKR\_RANDOM\_SEED\_NOT\_SUPPORTED 0x00000120UL
- #define CKR\_RANDOM\_NO\_RNG 0x00000121UL
- #define CKR\_DOMAIN\_PARAMS\_INVALID 0x00000130UL
- #define CKR\_CURVE\_NOT\_SUPPORTED 0x00000140UL
- #define CKR\_BUFFER\_TOO\_SMALL 0x00000150UL
- #define CKR\_SAVED\_STATE\_INVALID 0x00000160UL
- #define CKR\_INFORMATION\_SENSITIVE 0x00000170UL
- #define CKR\_STATE\_UNSAVEABLE 0x00000180UL
- #define CKR\_CRYPTOKI\_NOT\_INITIALIZED 0x00000190UL
- #define CKR\_CRYPTOKI\_ALREADY\_INITIALIZED 0x00000191UL
- #define CKR\_MUTEX\_BAD 0x000001A0UL
- #define CKR\_MUTEX\_NOT\_LOCKED 0x000001A1UL

- #define CKR\_NEW\_PIN\_MODE 0x000001B0UL
- #define CKR\_NEXT\_OTP 0x000001B1UL
- #define CKR\_EXCEEDED\_MAX\_ITERATIONS 0x000001B5UL
- #define CKR\_FIPS\_SELF\_TEST\_FAILED 0x000001B6UL
- #define CKR\_LIBRARY\_LOAD\_FAILED 0x000001B7UL
- #define CKR\_PIN\_TOO\_WEAK 0x000001B8UL
- #define CKR\_PUBLIC\_KEY\_INVALID 0x000001B9UL
- #define CKR\_FUNCTION\_REJECTED 0x00000200UL
- #define CKR\_VENDOR\_DEFINED 0x80000000UL
- #define CKF\_LIBRARY\_CANT\_CREATE\_OS\_THREADS 0x00000001UL
- #define CKF\_OS\_LOCKING\_OK 0x00000002UL
- #define CKF\_DONT\_BLOCK 1
- #define CKG\_MGF1\_SHA1 0x00000001UL
- #define CKG\_MGF1\_SHA256 0x00000002UL
- #define CKG\_MGF1\_SHA384 0x00000003UL
- #define CKG\_MGF1\_SHA512 0x00000004UL
- #define CKG\_MGF1\_SHA224 0x00000005UL
- #define CKZ\_DATA\_SPECIFIED 0x00000001UL
- #define CKD\_NULL 0x00000001UL
- #define CKD\_SHA1\_KDF 0x00000002UL
- #define CKD\_SHA1\_KDF\_ASN1 0x00000003UL
- #define CKD\_SHA1\_KDF\_CONCATENATE 0x00000004UL
- #define CKD\_SHA224\_KDF 0x00000005UL
- #define CKD\_SHA256\_KDF 0x00000006UL
- #define CKD\_SHA384\_KDF 0x00000007UL
- #define CKD\_SHA512\_KDF 0x00000008UL
- #define CKD\_CPDIVERSIFY\_KDF 0x00000009UL
- #define CKP\_PKCS5\_PBKD2\_HMAC\_SHA1 0x00000001UL
- #define CKP\_PKCS5\_PBKD2\_HMAC\_GOSTR3411 0x00000002UL
- #define CKP\_PKCS5\_PBKD2\_HMAC\_SHA224 0x00000003UL
- #define CKP\_PKCS5\_PBKD2\_HMAC\_SHA256 0x00000004UL
- #define CKP\_PKCS5\_PBKD2\_HMAC\_SHA384 0x00000005UL
- #define CKP\_PKCS5\_PBKD2\_HMAC\_SHA512 0x00000006UL
- #define CKP\_PKCS5\_PBKD2\_HMAC\_SHA512\_224 0x00000007UL
- #define CKP\_PKCS5\_PBKD2\_HMAC\_SHA512\_256 0x00000008UL
- #define CKZ\_SALT\_SPECIFIED 0x00000001UL
- #define CK\_OTP\_VALUE 0UL
- #define CK\_OTP\_PIN 1UL
- #define CK\_OTP\_CHALLENGE 2UL
- #define CK\_OTP\_TIME 3UL
- #define CK\_OTP\_COUNTER 4UL
- #define CK\_OTP\_FLAGS 5UL
- #define CK\_OTP\_OUTPUT\_LENGTH 6UL
- #define CK\_OTP\_OUTPUT\_FORMAT 7UL
- #define CKF\_NEXT\_OTP 0x00000001UL
- #define CKF\_EXCLUDE\_TIME 0x00000002UL
- #define CKF\_EXCLUDE\_COUNTER 0x00000004UL
- #define CKF\_EXCLUDE\_CHALLENGE 0x00000008UL
- #define CKF\_EXCLUDE\_PIN 0x00000010UL
- #define CKF\_USER\_FRIENDLY\_OTP 0x00000020UL

## Typedefs

- typedef unsigned char [CK\\_BYTE](#)
- typedef [CK\\_BYTE](#) [CK\\_CHAR](#)
- typedef [CK\\_BYTE](#) [CK\\_UTF8CHAR](#)
- typedef [CK\\_BYTE](#) [CK\\_BBOOL](#)
- typedef unsigned long int [CK\\_ULONG](#)
- typedef long int [CK\\_LONG](#)
- typedef [CK\\_ULONG](#) [CK\\_FLAGS](#)
- typedef [CK\\_BYTE](#) [CK\\_PTR](#) [CK\\_BYTE\\_PTR](#)
- typedef [CK\\_CHAR](#) [CK\\_PTR](#) [CK\\_CHAR\\_PTR](#)
- typedef [CK\\_UTF8CHAR](#) [CK\\_PTR](#) [CK\\_UTF8CHAR\\_PTR](#)
- typedef [CK\\_ULONG](#) [CK\\_PTR](#) [CK\\_ULONG\\_PTR](#)
- typedef void [CK\\_PTR](#) [CK\\_VOID\\_PTR](#)
- typedef [CK\\_VOID\\_PTR](#) [CK\\_PTR](#) [CK\\_VOID\\_PTR\\_PTR](#)
- typedef struct [CK\\_VERSION](#) [CK\\_VERSION](#)
- typedef [CK\\_VERSION](#) [CK\\_PTR](#) [CK\\_VERSION\\_PTR](#)
- typedef struct [CK\\_INFO](#) [CK\\_INFO](#)
- typedef [CK\\_INFO](#) [CK\\_PTR](#) [CK\\_INFO\\_PTR](#)
- typedef [CK\\_ULONG](#) [CK\\_NOTIFICATION](#)
- typedef [CK\\_ULONG](#) [CK\\_SLOT\\_ID](#)
- typedef [CK\\_SLOT\\_ID](#) [CK\\_PTR](#) [CK\\_SLOT\\_ID\\_PTR](#)
- typedef struct [CK\\_SLOT\\_INFO](#) [CK\\_SLOT\\_INFO](#)
- typedef [CK\\_SLOT\\_INFO](#) [CK\\_PTR](#) [CK\\_SLOT\\_INFO\\_PTR](#)
- typedef struct [CK\\_TOKEN\\_INFO](#) [CK\\_TOKEN\\_INFO](#)
- typedef [CK\\_TOKEN\\_INFO](#) [CK\\_PTR](#) [CK\\_TOKEN\\_INFO\\_PTR](#)
- typedef [CK\\_ULONG](#) [CK\\_SESSION\\_HANDLE](#)
- typedef [CK\\_SESSION\\_HANDLE](#) [CK\\_PTR](#) [CK\\_SESSION\\_HANDLE\\_PTR](#)
- typedef [CK\\_ULONG](#) [CK\\_USER\\_TYPE](#)
- typedef [CK\\_ULONG](#) [CK\\_STATE](#)
- typedef struct [CK\\_SESSION\\_INFO](#) [CK\\_SESSION\\_INFO](#)
- typedef [CK\\_SESSION\\_INFO](#) [CK\\_PTR](#) [CK\\_SESSION\\_INFO\\_PTR](#)
- typedef [CK\\_ULONG](#) [CK\\_OBJECT\\_HANDLE](#)
- typedef [CK\\_OBJECT\\_HANDLE](#) [CK\\_PTR](#) [CK\\_OBJECT\\_HANDLE\\_PTR](#)
- typedef [CK\\_ULONG](#) [CK\\_OBJECT\\_CLASS](#)
- typedef [CK\\_OBJECT\\_CLASS](#) [CK\\_PTR](#) [CK\\_OBJECT\\_CLASS\\_PTR](#)
- typedef [CK\\_ULONG](#) [CK\\_HW\\_FEATURE\\_TYPE](#)
- typedef [CK\\_ULONG](#) [CK\\_KEY\\_TYPE](#)
- typedef [CK\\_ULONG](#) [CK\\_CERTIFICATE\\_TYPE](#)
- typedef [CK\\_ULONG](#) [CK\\_ATTRIBUTE\\_TYPE](#)
- typedef struct [CK\\_ATTRIBUTE](#) [CK\\_ATTRIBUTE](#)
- typedef [CK\\_ATTRIBUTE](#) [CK\\_PTR](#) [CK\\_ATTRIBUTE\\_PTR](#)
- typedef struct [CK\\_DATE](#) [CK\\_DATE](#)
- typedef [CK\\_ULONG](#) [CK\\_MECHANISM\\_TYPE](#)
- typedef [CK\\_MECHANISM\\_TYPE](#) [CK\\_PTR](#) [CK\\_MECHANISM\\_TYPE\\_PTR](#)
- typedef struct [CK\\_MECHANISM](#) [CK\\_MECHANISM](#)
- typedef [CK\\_MECHANISM](#) [CK\\_PTR](#) [CK\\_MECHANISM\\_PTR](#)
- typedef struct [CK\\_MECHANISM\\_INFO](#) [CK\\_MECHANISM\\_INFO](#)
- typedef [CK\\_MECHANISM\\_INFO](#) [CK\\_PTR](#) [CK\\_MECHANISM\\_INFO\\_PTR](#)
- typedef [CK\\_ULONG](#) [CK\\_RV](#)
- typedef [CK\\_NOTIFICATION](#) event
- typedef [CK\\_NOTIFICATION](#) [CK\\_VOID\\_PTR](#) pApplication
- typedef struct [CK\\_FUNCTION\\_LIST](#) [CK\\_FUNCTION\\_LIST](#)
- typedef [CK\\_FUNCTION\\_LIST](#) [CK\\_PTR](#) [CK\\_FUNCTION\\_LIST\\_PTR](#)
- typedef [CK\\_FUNCTION\\_LIST\\_PTR](#) [CK\\_PTR](#) [CK\\_FUNCTION\\_LIST\\_PTR\\_PTR](#)

- typedef struct CK\_C\_INITIALIZE\_ARGS CK\_C\_INITIALIZE\_ARGS
- typedef CK\_C\_INITIALIZE\_ARGS CK\_PTR CK\_C\_INITIALIZE\_ARGS\_PTR
- typedef CK\_ULONG CK\_RSA\_PKCS\_MGF\_TYPE
- typedef CK\_RSA\_PKCS\_MGF\_TYPE CK\_PTR CK\_RSA\_PKCS\_MGF\_TYPE\_PTR
- typedef CK\_ULONG CK\_RSA\_PKCS\_OAEP\_SOURCE\_TYPE
- typedef CK\_RSA\_PKCS\_OAEP\_SOURCE\_TYPE CK\_PTR CK\_RSA\_PKCS\_OAEP\_SOURCE\_TYPE\_PTR
- typedef struct CK\_RSA\_PKCS\_OAEP\_PARAMS CK\_RSA\_PKCS\_OAEP\_PARAMS
- typedef CK\_RSA\_PKCS\_OAEP\_PARAMS CK\_PTR CK\_RSA\_PKCS\_OAEP\_PARAMS\_PTR
- typedef struct CK\_RSA\_PKCS\_PSS\_PARAMS CK\_RSA\_PKCS\_PSS\_PARAMS
- typedef CK\_RSA\_PKCS\_PSS\_PARAMS CK\_PTR CK\_RSA\_PKCS\_PSS\_PARAMS\_PTR
- typedef CK\_ULONG CK\_EC\_KDF\_TYPE
- typedef struct CK\_ECDH1\_DERIVE\_PARAMS CK\_ECDH1\_DERIVE\_PARAMS
- typedef CK\_ECDH1\_DERIVE\_PARAMS CK\_PTR CK\_ECDH1\_DERIVE\_PARAMS\_PTR
- typedef struct CK\_ECDH2\_DERIVE\_PARAMS CK\_ECDH2\_DERIVE\_PARAMS
- typedef CK\_ECDH2\_DERIVE\_PARAMS CK\_PTR CK\_ECDH2\_DERIVE\_PARAMS\_PTR
- typedef struct CK\_ECMQV\_DERIVE\_PARAMS CK\_ECMQV\_DERIVE\_PARAMS
- typedef CK\_ECMQV\_DERIVE\_PARAMS CK\_PTR CK\_ECMQV\_DERIVE\_PARAMS\_PTR
- typedef CK\_ULONG CK\_X9\_42\_DH\_KDF\_TYPE
- typedef CK\_X9\_42\_DH\_KDF\_TYPE CK\_PTR CK\_X9\_42\_DH\_KDF\_TYPE\_PTR
- typedef struct CK\_X9\_42\_DH1\_DERIVE\_PARAMS CK\_X9\_42\_DH1\_DERIVE\_PARAMS
- typedef struct CK\_X9\_42\_DH1\_DERIVE\_PARAMS CK\_PTR CK\_X9\_42\_DH1\_DERIVE\_PARAMS\_PTR
- typedef struct CK\_X9\_42\_DH2\_DERIVE\_PARAMS CK\_X9\_42\_DH2\_DERIVE\_PARAMS
- typedef CK\_X9\_42\_DH2\_DERIVE\_PARAMS CK\_PTR CK\_X9\_42\_DH2\_DERIVE\_PARAMS\_PTR
- typedef struct CK\_X9\_42\_MQV\_DERIVE\_PARAMS CK\_X9\_42\_MQV\_DERIVE\_PARAMS
- typedef CK\_X9\_42\_MQV\_DERIVE\_PARAMS CK\_PTR CK\_X9\_42\_MQV\_DERIVE\_PARAMS\_PTR
- typedef struct CK\_KEY\_DERIVE\_PARAMS CK\_KEY\_DERIVE\_PARAMS
- typedef CK\_KEY\_DERIVE\_PARAMS CK\_PTR CK\_KEY\_DERIVE\_PARAMS\_PTR
- typedef CK\_ULONG CK\_RC2\_PARAMS
- typedef CK\_RC2\_PARAMS CK\_PTR CK\_RC2\_PARAMS\_PTR
- typedef struct CK\_RC2\_CBC\_PARAMS CK\_RC2\_CBC\_PARAMS
- typedef CK\_RC2\_CBC\_PARAMS CK\_PTR CK\_RC2\_CBC\_PARAMS\_PTR
- typedef struct CK\_RC2\_MAC\_GENERAL\_PARAMS CK\_RC2\_MAC\_GENERAL\_PARAMS
- typedef CK\_RC2\_MAC\_GENERAL\_PARAMS CK\_PTR CK\_RC2\_MAC\_GENERAL\_PARAMS\_PTR
- typedef struct CK\_RC5\_PARAMS CK\_RC5\_PARAMS
- typedef CK\_RC5\_PARAMS CK\_PTR CK\_RC5\_PARAMS\_PTR
- typedef struct CK\_RC5\_CBC\_PARAMS CK\_RC5\_CBC\_PARAMS
- typedef CK\_RC5\_CBC\_PARAMS CK\_PTR CK\_RC5\_CBC\_PARAMS\_PTR
- typedef struct CK\_RC5\_MAC\_GENERAL\_PARAMS CK\_RC5\_MAC\_GENERAL\_PARAMS
- typedef CK\_RC5\_MAC\_GENERAL\_PARAMS CK\_PTR CK\_RC5\_MAC\_GENERAL\_PARAMS\_PTR
- typedef CK\_ULONG CK\_MAC\_GENERAL\_PARAMS
- typedef CK\_MAC\_GENERAL\_PARAMS CK\_PTR CK\_MAC\_GENERAL\_PARAMS\_PTR
- typedef struct CK\_DES\_CBC\_ENCRYPT\_DATA\_PARAMS CK\_DES\_CBC\_ENCRYPT\_DATA\_PARAMS
- typedef CK\_DES\_CBC\_ENCRYPT\_DATA\_PARAMS CK\_PTR CK\_DES\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR
- typedef struct CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS
- typedef CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS CK\_PTR CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR
- typedef struct CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS
- typedef CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS CK\_PTR CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS\_PTR
- typedef struct CK\_SKIPJACK\_RELAYX\_PARAMS CK\_SKIPJACK\_RELAYX\_PARAMS
- typedef CK\_SKIPJACK\_RELAYX\_PARAMS CK\_PTR CK\_SKIPJACK\_RELAYX\_PARAMS\_PTR
- typedef struct CK\_PBE\_PARAMS CK\_PBE\_PARAMS
- typedef CK\_PBE\_PARAMS CK\_PTR CK\_PBE\_PARAMS\_PTR
- typedef struct CK\_KEY\_WRAP\_SET\_OAEP\_PARAMS CK\_KEY\_WRAP\_SET\_OAEP\_PARAMS
- typedef CK\_KEY\_WRAP\_SET\_OAEP\_PARAMS CK\_PTR CK\_KEY\_WRAP\_SET\_OAEP\_PARAMS\_PTR
- typedef struct CK\_SSL3\_RANDOM\_DATA CK\_SSL3\_RANDOM\_DATA
- typedef struct CK\_SSL3\_MASTER\_KEY\_DERIVE\_PARAMS CK\_SSL3\_MASTER\_KEY\_DERIVE\_PARAMS

- typedef struct CK\_SSL3\_MASTER\_KEY\_DERIVE\_PARAMS CK\_PTR CK\_SSL3\_MASTER\_KEY\_DERIVE\_PARAMS\_PTR
- typedef struct CK\_SSL3\_KEY\_MAT\_OUT CK\_SSL3\_KEY\_MAT\_OUT
- typedef CK\_SSL3\_KEY\_MAT\_OUT CK\_PTR CK\_SSL3\_KEY\_MAT\_OUT\_PTR
- typedef struct CK\_SSL3\_KEY\_MAT\_PARAMS CK\_SSL3\_KEY\_MAT\_PARAMS
- typedef CK\_SSL3\_KEY\_MAT\_PARAMS CK\_PTR CK\_SSL3\_KEY\_MAT\_PARAMS\_PTR
- typedef struct CK\_TLS\_PRF\_PARAMS CK\_TLS\_PRF\_PARAMS
- typedef CK\_TLS\_PRF\_PARAMS CK\_PTR CK\_TLS\_PRF\_PARAMS\_PTR
- typedef struct CK\_WTLS\_RANDOM\_DATA CK\_WTLS\_RANDOM\_DATA
- typedef CK\_WTLS\_RANDOM\_DATA CK\_PTR CK\_WTLS\_RANDOM\_DATA\_PTR
- typedef struct CK\_WTLS\_MASTER\_KEY\_DERIVE\_PARAMS CK\_WTLS\_MASTER\_KEY\_DERIVE\_PARAMS
- typedef CK\_WTLS\_MASTER\_KEY\_DERIVE\_PARAMS CK\_PTR CK\_WTLS\_MASTER\_KEY\_DERIVE\_PARAMS\_PTR
- typedef struct CK\_WTLS\_PRF\_PARAMS CK\_WTLS\_PRF\_PARAMS
- typedef CK\_WTLS\_PRF\_PARAMS CK\_PTR CK\_WTLS\_PRF\_PARAMS\_PTR
- typedef struct CK\_WTLS\_KEY\_MAT\_OUT CK\_WTLS\_KEY\_MAT\_OUT
- typedef CK\_WTLS\_KEY\_MAT\_OUT CK\_PTR CK\_WTLS\_KEY\_MAT\_OUT\_PTR
- typedef struct CK\_WTLS\_KEY\_MAT\_PARAMS CK\_WTLS\_KEY\_MAT\_PARAMS
- typedef CK\_WTLS\_KEY\_MAT\_PARAMS CK\_PTR CK\_WTLS\_KEY\_MAT\_PARAMS\_PTR
- typedef struct CK\_CMS\_SIG\_PARAMS CK\_CMS\_SIG\_PARAMS
- typedef CK\_CMS\_SIG\_PARAMS CK\_PTR CK\_CMS\_SIG\_PARAMS\_PTR
- typedef struct CK\_KEY\_DERIVATION\_STRING\_DATA CK\_KEY\_DERIVATION\_STRING\_DATA
- typedef CK\_KEY\_DERIVATION\_STRING\_DATA CK\_PTR CK\_KEY\_DERIVATION\_STRING\_DATA\_PTR
- typedef CK\_ULONG CK\_EXTRACT\_PARAMS
- typedef CK\_EXTRACT\_PARAMS CK\_PTR CK\_EXTRACT\_PARAMS\_PTR
- typedef CK\_ULONG CK\_PKCS5\_PBKD2\_PSEUDO\_RANDOM\_FUNCTION\_TYPE
- typedef CK\_PKCS5\_PBKD2\_PSEUDO\_RANDOM\_FUNCTION\_TYPE CK\_PTR CK\_PKCS5\_PBKD2\_PSEUDO\_RANDOM\_F
- typedef CK\_ULONG CK\_PKCS5\_PBKDF2\_SALT\_SOURCE\_TYPE
- typedef CK\_PKCS5\_PBKDF2\_SALT\_SOURCE\_TYPE CK\_PTR CK\_PKCS5\_PBKDF2\_SALT\_SOURCE\_TYPE\_PTR
- typedef struct CK\_PKCS5\_PBKD2\_PARAMS CK\_PKCS5\_PBKD2\_PARAMS
- typedef CK\_PKCS5\_PBKD2\_PARAMS CK\_PTR CK\_PKCS5\_PBKD2\_PARAMS\_PTR
- typedef struct CK\_PKCS5\_PBKD2\_PARAMS2 CK\_PKCS5\_PBKD2\_PARAMS2
- typedef CK\_PKCS5\_PBKD2\_PARAMS2 CK\_PTR CK\_PKCS5\_PBKD2\_PARAMS2\_PTR
- typedef CK\_ULONG CK\_OTP\_PARAM\_TYPE
- typedef CK\_OTP\_PARAM\_TYPE CK\_PARAM\_TYPE
- typedef struct CK\_OTP\_PARAM CK\_OTP\_PARAM
- typedef CK\_OTP\_PARAM CK\_PTR CK\_OTP\_PARAM\_PTR
- typedef struct CK\_OTP\_PARAMS CK\_OTP\_PARAMS
- typedef CK\_OTP\_PARAMS CK\_PTR CK\_OTP\_PARAMS\_PTR
- typedef struct CK\_OTP\_SIGNATURE\_INFO CK\_OTP\_SIGNATURE\_INFO
- typedef CK\_OTP\_SIGNATURE\_INFO CK\_PTR CK\_OTP\_SIGNATURE\_INFO\_PTR
- typedef struct CK\_KIP\_PARAMS CK\_KIP\_PARAMS
- typedef CK\_KIP\_PARAMS CK\_PTR CK\_KIP\_PARAMS\_PTR
- typedef struct CK\_AES\_CTR\_PARAMS CK\_AES\_CTR\_PARAMS
- typedef CK\_AES\_CTR\_PARAMS CK\_PTR CK\_AES\_CTR\_PARAMS\_PTR
- typedef struct CK\_GCM\_PARAMS CK\_GCM\_PARAMS
- typedef CK\_GCM\_PARAMS CK\_PTR CK\_GCM\_PARAMS\_PTR
- typedef struct CK\_CCM\_PARAMS CK\_CCM\_PARAMS
- typedef CK\_CCM\_PARAMS CK\_PTR CK\_CCM\_PARAMS\_PTR
- typedef struct CK\_AES\_GCM\_PARAMS CK\_AES\_GCM\_PARAMS
- typedef CK\_AES\_GCM\_PARAMS CK\_PTR CK\_AES\_GCM\_PARAMS\_PTR
- typedef struct CK\_AES\_CCM\_PARAMS CK\_AES\_CCM\_PARAMS
- typedef CK\_AES\_CCM\_PARAMS CK\_PTR CK\_AES\_CCM\_PARAMS\_PTR
- typedef struct CK\_CAMELLIA\_CTR\_PARAMS CK\_CAMELLIA\_CTR\_PARAMS
- typedef CK\_CAMELLIA\_CTR\_PARAMS CK\_PTR CK\_CAMELLIA\_CTR\_PARAMS\_PTR
- typedef struct CK\_CAMELLIA\_CBC\_ENCRYPT\_DATA\_PARAMS CK\_CAMELLIA\_CBC\_ENCRYPT\_DATA\_PARAMS
- typedef CK\_CAMELLIA\_CBC\_ENCRYPT\_DATA\_PARAMS CK\_PTR CK\_CAMELLIA\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR

- typedef struct CK\_ARIA\_CBC\_ENCRYPT\_DATA\_PARAMS CK\_ARIA\_CBC\_ENCRYPT\_DATA\_PARAMS
- typedef CK\_ARIA\_CBC\_ENCRYPT\_DATA\_PARAMS CK\_PTR CK\_ARIA\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR
- typedef struct CK\_DSA\_PARAMETER\_GEN\_PARAM CK\_DSA\_PARAMETER\_GEN\_PARAM
- typedef CK\_DSA\_PARAMETER\_GEN\_PARAM CK\_PTR CK\_DSA\_PARAMETER\_GEN\_PARAM\_PTR
- typedef struct CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS
- typedef CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS CK\_PTR CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS\_PTR
- typedef CK\_ULONG CK\_JAVA\_MIDP\_SECURITY\_DOMAIN
- typedef CK\_ULONG CK\_CERTIFICATE\_CATEGORY
- typedef struct CK\_RSA\_AES\_KEY\_WRAP\_PARAMS CK\_RSA\_AES\_KEY\_WRAP\_PARAMS
- typedef CK\_RSA\_AES\_KEY\_WRAP\_PARAMS CK\_PTR CK\_RSA\_AES\_KEY\_WRAP\_PARAMS\_PTR
- typedef struct CK\_TLS12\_MASTER\_KEY\_DERIVE\_PARAMS CK\_TLS12\_MASTER\_KEY\_DERIVE\_PARAMS
- typedef CK\_TLS12\_MASTER\_KEY\_DERIVE\_PARAMS CK\_PTR CK\_TLS12\_MASTER\_KEY\_DERIVE\_PARAMS\_PTR
- typedef struct CK\_TLS12\_KEY\_MAT\_PARAMS CK\_TLS12\_KEY\_MAT\_PARAMS
- typedef CK\_TLS12\_KEY\_MAT\_PARAMS CK\_PTR CK\_TLS12\_KEY\_MAT\_PARAMS\_PTR
- typedef struct CK\_TLS\_KDF\_PARAMS CK\_TLS\_KDF\_PARAMS
- typedef CK\_TLS\_KDF\_PARAMS CK\_PTR CK\_TLS\_KDF\_PARAMS\_PTR
- typedef struct CK\_TLS\_MAC\_PARAMS CK\_TLS\_MAC\_PARAMS
- typedef CK\_TLS\_MAC\_PARAMS CK\_PTR CK\_TLS\_MAC\_PARAMS\_PTR
- typedef struct CK\_GOSTR3410\_DERIVE\_PARAMS CK\_GOSTR3410\_DERIVE\_PARAMS
- typedef CK\_GOSTR3410\_DERIVE\_PARAMS CK\_PTR CK\_GOSTR3410\_DERIVE\_PARAMS\_PTR
- typedef struct CK\_GOSTR3410\_KEY\_WRAP\_PARAMS CK\_GOSTR3410\_KEY\_WRAP\_PARAMS
- typedef CK\_GOSTR3410\_KEY\_WRAP\_PARAMS CK\_PTR CK\_GOSTR3410\_KEY\_WRAP\_PARAMS\_PTR
- typedef struct CK\_SEED\_CBC\_ENCRYPT\_DATA\_PARAMS CK\_SEED\_CBC\_ENCRYPT\_DATA\_PARAMS
- typedef CK\_SEED\_CBC\_ENCRYPT\_DATA\_PARAMS CK\_PTR CK\_SEED\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR

## Functions

- typedef CK\_CALLBACK\_FUNCTION (CK\_RV, CK\_NOTIFY)(CK\_SESSION\_HANDLE hSession
- typedef CK\_CALLBACK\_FUNCTION (CK\_RV, CK\_CREATEMUTEX)(CK\_VOID\_PTR\_PTR ppMutex)
- typedef CK\_CALLBACK\_FUNCTION (CK\_RV, CK\_DESTROYMUTEX)(CK\_VOID\_PTR pMutex)
- typedef CK\_CALLBACK\_FUNCTION (CK\_RV, CK\_LOCKMUTEX)(CK\_VOID\_PTR pMutex)
- typedef CK\_CALLBACK\_FUNCTION (CK\_RV, CK\_UNLOCKMUTEX)(CK\_VOID\_PTR pMutex)

## 20.171.1 Macro Definition Documentation

### 20.171.1.1 CK\_CERTIFICATE\_CATEGORY\_AUTHORITY

```
#define CK_CERTIFICATE_CATEGORY_AUTHORITY 2UL
```

### 20.171.1.2 CK\_CERTIFICATE\_CATEGORY\_OTHER\_ENTITY

```
#define CK_CERTIFICATE_CATEGORY_OTHER_ENTITY 3UL
```

### 20.171.1.3 CK\_CERTIFICATE\_CATEGORY\_TOKEN\_USER

```
#define CK_CERTIFICATE_CATEGORY_TOKEN_USER 1UL
```

### 20.171.1.4 CK\_CERTIFICATE\_CATEGORY\_UNSPECIFIED

```
#define CK_CERTIFICATE_CATEGORY_UNSPECIFIED 0UL
```

### 20.171.1.5 CK\_EFFECTIVELY\_INFINITE

```
#define CK_EFFECTIVELY_INFINITE 0UL
```

### 20.171.1.6 CK\_FALSE

```
#define CK_FALSE 0
```

### 20.171.1.7 CK\_INVALID\_HANDLE

```
#define CK_INVALID_HANDLE 0UL
```

### 20.171.1.8 CK\_OTP\_CHALLENGE

```
#define CK_OTP_CHALLENGE 2UL
```

### 20.171.1.9 CK\_OTP\_COUNTER

```
#define CK_OTP_COUNTER 4UL
```

### 20.171.1.10 CK\_OTP\_FLAGS

```
#define CK_OTP_FLAGS 5UL
```



**20.171.1.11 CK\_OTP\_FORMAT\_ALPHANUMERIC**

```
#define CK_OTP_FORMAT_ALPHANUMERIC 2UL
```

**20.171.1.12 CK\_OTP\_FORMAT\_BINARY**

```
#define CK_OTP_FORMAT_BINARY 3UL
```

**20.171.1.13 CK\_OTP\_FORMAT\_DECIMAL**

```
#define CK_OTP_FORMAT_DECIMAL 0UL
```

**20.171.1.14 CK\_OTP\_FORMAT\_HEXADECIMAL**

```
#define CK_OTP_FORMAT_HEXADECIMAL 1UL
```

**20.171.1.15 CK\_OTP\_OUTPUT\_FORMAT**

```
#define CK_OTP_OUTPUT_FORMAT 7UL
```

**20.171.1.16 CK\_OTP\_OUTPUT\_LENGTH**

```
#define CK_OTP_OUTPUT_LENGTH 6UL
```

**20.171.1.17 CK\_OTP\_PARAM\_IGNORED**

```
#define CK_OTP_PARAM_IGNORED 0UL
```

**20.171.1.18 CK\_OTP\_PARAM\_MANDATORY**

```
#define CK_OTP_PARAM_MANDATORY 2UL
```

### 20.171.1.19 CK\_OTP\_PARAM\_OPTIONAL

```
#define CK_OTP_PARAM_OPTIONAL 1UL
```

### 20.171.1.20 CK\_OTP\_PIN

```
#define CK_OTP_PIN 1UL
```

### 20.171.1.21 CK\_OTP\_TIME

```
#define CK_OTP_TIME 3UL
```

### 20.171.1.22 CK\_OTP\_VALUE

```
#define CK_OTP_VALUE 0UL
```

### 20.171.1.23 CK\_SECURITY\_DOMAIN\_MANUFACTURER

```
#define CK_SECURITY_DOMAIN_MANUFACTURER 1UL
```

### 20.171.1.24 CK\_SECURITY\_DOMAIN\_OPERATOR

```
#define CK_SECURITY_DOMAIN_OPERATOR 2UL
```

### 20.171.1.25 CK\_SECURITY\_DOMAIN\_THIRD\_PARTY

```
#define CK_SECURITY_DOMAIN_THIRD_PARTY 3UL
```

### 20.171.1.26 CK\_SECURITY\_DOMAIN\_UNSPECIFIED

```
#define CK_SECURITY_DOMAIN_UNSPECIFIED 0UL
```

**20.171.1.27 CK\_TRUE**

```
#define CK_TRUE 1
```

**20.171.1.28 CK\_UNAVAILABLE\_INFORMATION**

```
#define CK_UNAVAILABLE_INFORMATION (~0UL)
```

**20.171.1.29 CKA\_AC\_ISSUER**

```
#define CKA_AC_ISSUER 0x00000083UL
```

**20.171.1.30 CKA\_ALLOWED\_MECHANISMS**

```
#define CKA_ALLOWED_MECHANISMS (CKF_ARRAY_ATTRIBUTE | 0x00000600UL)
```

**20.171.1.31 CKA\_ALWAYS\_AUTHENTICATE**

```
#define CKA_ALWAYS_AUTHENTICATE 0x00000202UL
```

**20.171.1.32 CKA\_ALWAYS\_SENSITIVE**

```
#define CKA_ALWAYS_SENSITIVE 0x00000165UL
```

**20.171.1.33 CKA\_APPLICATION**

```
#define CKA_APPLICATION 0x00000010UL
```

**20.171.1.34 CKA\_ATTR\_TYPES**

```
#define CKA_ATTR_TYPES 0x00000085UL
```

### 20.171.1.35 CKA\_AUTH\_PIN\_FLAGS

```
#define CKA_AUTH_PIN_FLAGS 0x00000201UL /* Deprecated */
```

### 20.171.1.36 CKA\_BASE

```
#define CKA_BASE 0x00000132UL
```

### 20.171.1.37 CKA\_BITS\_PER\_PIXEL

```
#define CKA_BITS_PER_PIXEL 0x00000406UL
```

### 20.171.1.38 CKA\_CERTIFICATE\_CATEGORY

```
#define CKA_CERTIFICATE_CATEGORY 0x00000087UL
```

### 20.171.1.39 CKA\_CERTIFICATE\_TYPE

```
#define CKA_CERTIFICATE_TYPE 0x00000080UL
```

### 20.171.1.40 CKA\_CHAR\_COLUMNS

```
#define CKA_CHAR_COLUMNS 0x00000404UL
```

### 20.171.1.41 CKA\_CHAR\_ROWS

```
#define CKA_CHAR_ROWS 0x00000403UL
```

### 20.171.1.42 CKA\_CHAR\_SETS

```
#define CKA_CHAR_SETS 0x00000480UL
```

**20.171.1.43 CKA\_CHECK\_VALUE**

```
#define CKA_CHECK_VALUE 0x00000090UL
```

**20.171.1.44 CKA\_CLASS**

```
#define CKA_CLASS 0x00000000UL
```

**20.171.1.45 CKA\_COEFFICIENT**

```
#define CKA_COEFFICIENT 0x00000128UL
```

**20.171.1.46 CKA\_COLOR**

```
#define CKA_COLOR 0x00000405UL
```

**20.171.1.47 CKA\_COPYABLE**

```
#define CKA_COPYABLE 0x00000171UL
```

**20.171.1.48 CKA\_DECRYPT**

```
#define CKA_DECRYPT 0x00000105UL
```

**20.171.1.49 CKA\_DEFAULT\_CMS\_ATTRIBUTES**

```
#define CKA_DEFAULT_CMS_ATTRIBUTES 0x00000502UL
```

**20.171.1.50 CKA\_DERIVE**

```
#define CKA_DERIVE 0x0000010CUL
```

### 20.171.1.51 CKA\_DERIVE\_TEMPLATE

```
#define CKA_DERIVE_TEMPLATE (CKF_ARRAY_ATTRIBUTE | 0x00000213UL)
```

### 20.171.1.52 CKA\_DESTROYABLE

```
#define CKA_DESTROYABLE 0x00000172UL
```

### 20.171.1.53 CKA\_EC\_PARAMS

```
#define CKA_EC_PARAMS 0x00000180UL
```

### 20.171.1.54 CKA\_EC\_POINT

```
#define CKA_EC_POINT 0x00000181UL
```

### 20.171.1.55 CKA\_ECDSA\_PARAMS

```
#define CKA_ECDSA_PARAMS 0x00000180UL /* Deprecated */
```

### 20.171.1.56 CKA\_ENCODING\_METHODS

```
#define CKA_ENCODING_METHODS 0x00000481UL
```

### 20.171.1.57 CKA\_ENCRYPT

```
#define CKA_ENCRYPT 0x00000104UL
```

### 20.171.1.58 CKA\_END\_DATE

```
#define CKA_END_DATE 0x00000111UL
```

**20.171.1.59 CKA\_EXPONENT\_1**

```
#define CKA_EXPONENT_1 0x00000126UL
```

**20.171.1.60 CKA\_EXPONENT\_2**

```
#define CKA_EXPONENT_2 0x00000127UL
```

**20.171.1.61 CKA\_EXTRACTABLE**

```
#define CKA_EXTRACTABLE 0x00000162UL
```

**20.171.1.62 CKA\_GOST28147\_PARAMS**

```
#define CKA_GOST28147_PARAMS 0x00000252UL
```

**20.171.1.63 CKA\_GOSTR3410\_PARAMS**

```
#define CKA_GOSTR3410_PARAMS 0x00000250UL
```

**20.171.1.64 CKA\_GOSTR3411\_PARAMS**

```
#define CKA_GOSTR3411_PARAMS 0x00000251UL
```

**20.171.1.65 CKA\_HAS\_RESET**

```
#define CKA_HAS_RESET 0x00000302UL
```

**20.171.1.66 CKA\_HASH\_OF\_ISSUER\_PUBLIC\_KEY**

```
#define CKA_HASH_OF_ISSUER_PUBLIC_KEY 0x0000008BUL
```

### 20.171.1.67 CKA\_HASH\_OF\_SUBJECT\_PUBLIC\_KEY

```
#define CKA_HASH_OF_SUBJECT_PUBLIC_KEY 0x0000008AUL
```

### 20.171.1.68 CKA\_HW\_FEATURE\_TYPE

```
#define CKA_HW_FEATURE_TYPE 0x00000300UL
```

### 20.171.1.69 CKA\_ID

```
#define CKA_ID 0x00000102UL
```

### 20.171.1.70 CKA\_ISSUER

```
#define CKA_ISSUER 0x00000081UL
```

### 20.171.1.71 CKA\_JAVA\_MIDP\_SECURITY\_DOMAIN

```
#define CKA_JAVA_MIDP_SECURITY_DOMAIN 0x00000088UL
```

### 20.171.1.72 CKA\_KEY\_GEN\_MECHANISM

```
#define CKA_KEY_GEN_MECHANISM 0x00000166UL
```

### 20.171.1.73 CKA\_KEY\_TYPE

```
#define CKA_KEY_TYPE 0x00000100UL
```

### 20.171.1.74 CKA\_LABEL

```
#define CKA_LABEL 0x00000003UL
```



**20.171.1.75 CKA\_LOCAL**

```
#define CKA_LOCAL 0x00000163UL
```

**20.171.1.76 CKA\_MECHANISM\_TYPE**

```
#define CKA_MECHANISM_TYPE 0x00000500UL
```

**20.171.1.77 CKA\_MIME\_TYPES**

```
#define CKA_MIME_TYPES 0x00000482UL
```

**20.171.1.78 CKA\_MODIFIABLE**

```
#define CKA_MODIFIABLE 0x00000170UL
```

**20.171.1.79 CKA\_MODULUS**

```
#define CKA_MODULUS 0x00000120UL
```

**20.171.1.80 CKA\_MODULUS\_BITS**

```
#define CKA_MODULUS_BITS 0x00000121UL
```

**20.171.1.81 CKA\_NAME\_HASH\_ALGORITHM**

```
#define CKA_NAME_HASH_ALGORITHM 0x0000008CUL
```

**20.171.1.82 CKA\_NEVER\_EXTRACTABLE**

```
#define CKA_NEVER_EXTRACTABLE 0x00000164UL
```

### 20.171.1.83 CKA\_OBJECT\_ID

```
#define CKA_OBJECT_ID 0x00000012UL
```

### 20.171.1.84 CKA\_OTP\_CHALLENGE\_REQUIREMENT

```
#define CKA_OTP_CHALLENGE_REQUIREMENT 0x00000224UL
```

### 20.171.1.85 CKA\_OTP\_COUNTER

```
#define CKA_OTP_COUNTER 0x0000022EUL
```

### 20.171.1.86 CKA\_OTP\_COUNTER\_REQUIREMENT

```
#define CKA_OTP_COUNTER_REQUIREMENT 0x00000226UL
```

### 20.171.1.87 CKA\_OTP\_FORMAT

```
#define CKA_OTP_FORMAT 0x00000220UL
```

### 20.171.1.88 CKA\_OTP\_LENGTH

```
#define CKA_OTP_LENGTH 0x00000221UL
```

### 20.171.1.89 CKA\_OTP\_PIN\_REQUIREMENT

```
#define CKA_OTP_PIN_REQUIREMENT 0x00000227UL
```

### 20.171.1.90 CKA\_OTP\_SERVICE\_IDENTIFIER

```
#define CKA_OTP_SERVICE_IDENTIFIER 0x0000022BUL
```

**20.171.1.91 CKA\_OTP\_SERVICE\_LOGO**

```
#define CKA_OTP_SERVICE_LOGO 0x0000022CUL
```

**20.171.1.92 CKA\_OTP\_SERVICE\_LOGO\_TYPE**

```
#define CKA_OTP_SERVICE_LOGO_TYPE 0x0000022DUL
```

**20.171.1.93 CKA\_OTP\_TIME**

```
#define CKA_OTP_TIME 0x0000022FUL
```

**20.171.1.94 CKA\_OTP\_TIME\_INTERVAL**

```
#define CKA_OTP_TIME_INTERVAL 0x00000222UL
```

**20.171.1.95 CKA\_OTP\_TIME\_REQUIREMENT**

```
#define CKA_OTP_TIME_REQUIREMENT 0x00000225UL
```

**20.171.1.96 CKA\_OTP\_USER\_FRIENDLY\_MODE**

```
#define CKA_OTP_USER_FRIENDLY_MODE 0x00000223UL
```

**20.171.1.97 CKA\_OTP\_USER\_IDENTIFIER**

```
#define CKA_OTP_USER_IDENTIFIER 0x0000022AUL
```

**20.171.1.98 CKA\_OWNER**

```
#define CKA_OWNER 0x00000084UL
```

### 20.171.1.99 CKA\_PIXEL\_X

```
#define CKA_PIXEL_X 0x00000400UL
```

### 20.171.1.100 CKA\_PIXEL\_Y

```
#define CKA_PIXEL_Y 0x00000401UL
```

### 20.171.1.101 CKA\_PRIME

```
#define CKA_PRIME 0x00000130UL
```

### 20.171.1.102 CKA\_PRIME\_1

```
#define CKA_PRIME_1 0x00000124UL
```

### 20.171.1.103 CKA\_PRIME\_2

```
#define CKA_PRIME_2 0x00000125UL
```

### 20.171.1.104 CKA\_PRIME\_BITS

```
#define CKA_PRIME_BITS 0x00000133UL
```

### 20.171.1.105 CKA\_PRIVATE

```
#define CKA_PRIVATE 0x00000002UL
```

### 20.171.1.106 CKA\_PRIVATE\_EXPONENT

```
#define CKA_PRIVATE_EXPONENT 0x00000123UL
```

**20.171.1.107 CKA\_PUBLIC\_EXPONENT**

```
#define CKA_PUBLIC_EXPONENT 0x00000122UL
```

**20.171.1.108 CKA\_PUBLIC\_KEY\_INFO**

```
#define CKA_PUBLIC_KEY_INFO 0x00000129UL
```

**20.171.1.109 CKA\_REQUIRED\_CMS\_ATTRIBUTES**

```
#define CKA_REQUIRED_CMS_ATTRIBUTES 0x00000501UL
```

**20.171.1.110 CKA\_RESET\_ON\_INIT**

```
#define CKA_RESET_ON_INIT 0x00000301UL
```

**20.171.1.111 CKA\_RESOLUTION**

```
#define CKA_RESOLUTION 0x00000402UL
```

**20.171.1.112 CKA\_SECONDARY\_AUTH**

```
#define CKA_SECONDARY_AUTH 0x00000200UL /* Deprecated */
```

**20.171.1.113 CKA\_SENSITIVE**

```
#define CKA_SENSITIVE 0x00000103UL
```

**20.171.1.114 CKA\_SERIAL\_NUMBER**

```
#define CKA_SERIAL_NUMBER 0x00000082UL
```

### 20.171.1.115 CKA\_SIGN

```
#define CKA_SIGN 0x00000108UL
```

### 20.171.1.116 CKA\_SIGN\_RECOVER

```
#define CKA_SIGN_RECOVER 0x00000109UL
```

### 20.171.1.117 CKA\_START\_DATE

```
#define CKA_START_DATE 0x00000110UL
```

### 20.171.1.118 CKA\_SUB\_PRIME\_BITS

```
#define CKA_SUB_PRIME_BITS CKA_SUBPRIME_BITS
```

### 20.171.1.119 CKA\_SUBJECT

```
#define CKA_SUBJECT 0x00000101UL
```

### 20.171.1.120 CKA\_SUBPRIME

```
#define CKA_SUBPRIME 0x00000131UL
```

### 20.171.1.121 CKA\_SUBPRIME\_BITS

```
#define CKA_SUBPRIME_BITS 0x00000134UL
```

### 20.171.1.122 CKA\_SUPPORTED\_CMS\_ATTRIBUTES

```
#define CKA_SUPPORTED_CMS_ATTRIBUTES 0x00000503UL
```

**20.171.1.123 CKA\_TOKEN**

```
#define CKA_TOKEN 0x00000001UL
```

**20.171.1.124 CKA\_TRUSTED**

```
#define CKA_TRUSTED 0x00000086UL
```

**20.171.1.125 CKA\_UNWRAP**

```
#define CKA_UNWRAP 0x00000107UL
```

**20.171.1.126 CKA\_UNWRAP\_TEMPLATE**

```
#define CKA_UNWRAP_TEMPLATE (CKF_ARRAY_ATTRIBUTE | 0x00000212UL)
```

**20.171.1.127 CKA\_URL**

```
#define CKA_URL 0x00000089UL
```

**20.171.1.128 CKA\_VALUE**

```
#define CKA_VALUE 0x00000011UL
```

**20.171.1.129 CKA\_VALUE\_BITS**

```
#define CKA_VALUE_BITS 0x00000160UL
```

**20.171.1.130 CKA\_VALUE\_LEN**

```
#define CKA_VALUE_LEN 0x00000161UL
```

### 20.171.1.131 CKA\_VENDOR\_DEFINED

```
#define CKA_VENDOR_DEFINED 0x80000000UL
```

### 20.171.1.132 CKA\_VERIFY

```
#define CKA_VERIFY 0x0000010AUL
```

### 20.171.1.133 CKA\_VERIFY\_RECOVER

```
#define CKA_VERIFY_RECOVER 0x0000010BUL
```

### 20.171.1.134 CKA\_WRAP

```
#define CKA_WRAP 0x00000106UL
```

### 20.171.1.135 CKA\_WRAP\_TEMPLATE

```
#define CKA_WRAP_TEMPLATE (CKF_ARRAY_ATTRIBUTE | 0x00000211UL)
```

### 20.171.1.136 CKA\_WRAP\_WITH\_TRUSTED

```
#define CKA_WRAP_WITH_TRUSTED 0x00000210UL
```

### 20.171.1.137 CKC\_OPENPGP

```
#define CKC_OPENPGP (CKC_VENDOR_DEFINED | 0x00504750)
```

### 20.171.1.138 CKC\_VENDOR\_DEFINED

```
#define CKC_VENDOR_DEFINED 0x80000000UL
```



**20.171.1.139 CKC\_WTLS**

```
#define CKC_WTLS 0x00000002UL
```

**20.171.1.140 CKC\_X\_509**

```
#define CKC_X_509 0x00000000UL
```

**20.171.1.141 CKC\_X\_509\_ATTR\_CERT**

```
#define CKC_X_509_ATTR_CERT 0x00000001UL
```

**20.171.1.142 CKD\_CPDIVERSIFY\_KDF**

```
#define CKD_CPDIVERSIFY_KDF 0x00000009UL
```

**20.171.1.143 CKD\_NULL**

```
#define CKD_NULL 0x00000001UL
```

**20.171.1.144 CKD\_SHA1\_KDF**

```
#define CKD_SHA1_KDF 0x00000002UL
```

**20.171.1.145 CKD\_SHA1\_KDF\_ASN1**

```
#define CKD_SHA1_KDF_ASN1 0x00000003UL
```

**20.171.1.146 CKD\_SHA1\_KDF\_CONCATENATE**

```
#define CKD_SHA1_KDF_CONCATENATE 0x00000004UL
```

### 20.171.1.147 CKD\_SHA224\_KDF

```
#define CKD_SHA224_KDF 0x00000005UL
```

### 20.171.1.148 CKD\_SHA256\_KDF

```
#define CKD_SHA256_KDF 0x00000006UL
```

### 20.171.1.149 CKD\_SHA384\_KDF

```
#define CKD_SHA384_KDF 0x00000007UL
```

### 20.171.1.150 CKD\_SHA512\_KDF

```
#define CKD_SHA512_KDF 0x00000008UL
```

### 20.171.1.151 CKF\_ARRAY\_ATTRIBUTE

```
#define CKF_ARRAY_ATTRIBUTE 0x40000000UL
```

### 20.171.1.152 CKF\_CLOCK\_ON\_TOKEN

```
#define CKF_CLOCK_ON_TOKEN 0x00000040UL
```

### 20.171.1.153 CKF\_DECRYPT

```
#define CKF_DECRYPT 0x00000200UL
```

### 20.171.1.154 CKF\_DERIVE

```
#define CKF_DERIVE 0x00080000UL
```

**20.171.1.155 CKF\_DIGEST**

```
#define CKF_DIGEST 0x00000400UL
```

**20.171.1.156 CKF\_DONT\_BLOCK**

```
#define CKF_DONT_BLOCK 1
```

**20.171.1.157 CKF\_DUAL\_CRYPTO\_OPERATIONS**

```
#define CKF_DUAL_CRYPTO_OPERATIONS 0x00000200UL
```

**20.171.1.158 CKF\_EC\_COMPRESS**

```
#define CKF_EC_COMPRESS 0x02000000UL
```

**20.171.1.159 CKF\_EC\_ECPARAMETERS**

```
#define CKF_EC_ECPARAMETERS 0x00400000UL
```

**20.171.1.160 CKF\_EC\_F\_2M**

```
#define CKF_EC_F_2M 0x00200000UL
```

**20.171.1.161 CKF\_EC\_F\_P**

```
#define CKF_EC_F_P 0x00100000UL
```

**20.171.1.162 CKF\_EC\_NAMEDCURVE**

```
#define CKF_EC_NAMEDCURVE 0x00800000UL
```

### 20.171.1.163 CKF\_EC\_UNCOMPRESS

```
#define CKF_EC_UNCOMPRESS 0x01000000UL
```

### 20.171.1.164 CKF\_ENCRYPT

```
#define CKF_ENCRYPT 0x00000100UL
```

### 20.171.1.165 CKF\_ERROR\_STATE

```
#define CKF_ERROR_STATE 0x01000000UL
```

### 20.171.1.166 CKF\_EXCLUDE\_CHALLENGE

```
#define CKF_EXCLUDE_CHALLENGE 0x00000008UL
```

### 20.171.1.167 CKF\_EXCLUDE\_COUNTER

```
#define CKF_EXCLUDE_COUNTER 0x00000004UL
```

### 20.171.1.168 CKF\_EXCLUDE\_PIN

```
#define CKF_EXCLUDE_PIN 0x00000010UL
```

### 20.171.1.169 CKF\_EXCLUDE\_TIME

```
#define CKF_EXCLUDE_TIME 0x00000002UL
```

### 20.171.1.170 CKF\_EXTENSION

```
#define CKF_EXTENSION 0x80000000UL
```

**20.171.1.171 CKF\_GENERATE**

```
#define CKF_GENERATE 0x00008000UL
```

**20.171.1.172 CKF\_GENERATE\_KEY\_PAIR**

```
#define CKF_GENERATE_KEY_PAIR 0x00010000UL
```

**20.171.1.173 CKF\_HW**

```
#define CKF_HW 0x00000001UL /* performed by HW */
```

**20.171.1.174 CKF\_HW\_SLOT**

```
#define CKF_HW_SLOT 0x00000004UL /* hardware slot */
```

**20.171.1.175 CKF\_LIBRARY\_CANT\_CREATE\_OS\_THREADS**

```
#define CKF_LIBRARY_CANT_CREATE_OS_THREADS 0x00000001UL
```

**20.171.1.176 CKF\_LOGIN\_REQUIRED**

```
#define CKF_LOGIN_REQUIRED 0x00000004UL /* user must login */
```

**20.171.1.177 CKF\_NEXT\_OTP**

```
#define CKF_NEXT_OTP 0x00000001UL
```

**20.171.1.178 CKF\_OS\_LOCKING\_OK**

```
#define CKF_OS_LOCKING_OK 0x00000002UL
```

### 20.171.1.179 CKF\_PROTECTED\_AUTHENTICATION\_PATH

```
#define CKF_PROTECTED_AUTHENTICATION_PATH 0x00000100UL
```

### 20.171.1.180 CKF\_REMOVABLE\_DEVICE

```
#define CKF_REMOVABLE_DEVICE 0x00000002UL /* removable devices*/
```

### 20.171.1.181 CKF\_RESTORE\_KEY\_NOT\_NEEDED

```
#define CKF_RESTORE_KEY_NOT_NEEDED 0x00000020UL
```

### 20.171.1.182 CKF\_RNG

```
#define CKF_RNG 0x00000001UL /* has random # generator */
```

### 20.171.1.183 CKF\_RW\_SESSION

```
#define CKF_RW_SESSION 0x00000002UL /* session is r/w */
```

### 20.171.1.184 CKF\_SECONDARY\_AUTHENTICATION

```
#define CKF_SECONDARY_AUTHENTICATION 0x00000800UL
```

### 20.171.1.185 CKF\_SERIAL\_SESSION

```
#define CKF_SERIAL_SESSION 0x00000004UL /* no parallel */
```

### 20.171.1.186 CKF\_SIGN

```
#define CKF_SIGN 0x00000800UL
```

**20.171.1.187 CKF\_SIGN\_RECOVER**

```
#define CKF_SIGN_RECOVER 0x00001000UL
```

**20.171.1.188 CKF\_SO\_PIN\_COUNT\_LOW**

```
#define CKF_SO_PIN_COUNT_LOW 0x00100000UL
```

**20.171.1.189 CKF\_SO\_PIN\_FINAL\_TRY**

```
#define CKF_SO_PIN_FINAL_TRY 0x00200000UL
```

**20.171.1.190 CKF\_SO\_PIN\_LOCKED**

```
#define CKF_SO_PIN_LOCKED 0x00400000UL
```

**20.171.1.191 CKF\_SO\_PIN\_TO\_BE\_CHANGED**

```
#define CKF_SO_PIN_TO_BE_CHANGED 0x00800000UL
```

**20.171.1.192 CKF\_TOKEN\_INITIALIZED**

```
#define CKF_TOKEN_INITIALIZED 0x00000400UL
```

**20.171.1.193 CKF\_TOKEN\_PRESENT**

```
#define CKF_TOKEN_PRESENT 0x00000001UL /* a token is there */
```

**20.171.1.194 CKF\_UNWRAP**

```
#define CKF_UNWRAP 0x00040000UL
```

### 20.171.1.195 CKF\_USER\_FRIENDLY\_OTP

```
#define CKF_USER_FRIENDLY_OTP 0x00000020UL
```

### 20.171.1.196 CKF\_USER\_PIN\_COUNT\_LOW

```
#define CKF_USER_PIN_COUNT_LOW 0x00010000UL
```

### 20.171.1.197 CKF\_USER\_PIN\_FINAL\_TRY

```
#define CKF_USER_PIN_FINAL_TRY 0x00020000UL
```

### 20.171.1.198 CKF\_USER\_PIN\_INITIALIZED

```
#define CKF_USER_PIN_INITIALIZED 0x00000008UL /* normal user's PIN is set */
```

### 20.171.1.199 CKF\_USER\_PIN\_LOCKED

```
#define CKF_USER_PIN_LOCKED 0x00040000UL
```

### 20.171.1.200 CKF\_USER\_PIN\_TO\_BE\_CHANGED

```
#define CKF_USER_PIN_TO_BE_CHANGED 0x00080000UL
```

### 20.171.1.201 CKF\_VERIFY

```
#define CKF_VERIFY 0x00002000UL
```

### 20.171.1.202 CKF\_VERIFY\_RECOVER

```
#define CKF_VERIFY_RECOVER 0x00004000UL
```



**20.171.1.203 CKF\_WRAP**

```
#define CKF_WRAP 0x00020000UL
```

**20.171.1.204 CKF\_WRITE\_PROTECTED**

```
#define CKF_WRITE_PROTECTED 0x00000002UL /* token is write-protected */
```

**20.171.1.205 CKG\_MGF1\_SHA1**

```
#define CKG_MGF1_SHA1 0x00000001UL
```

**20.171.1.206 CKG\_MGF1\_SHA224**

```
#define CKG_MGF1_SHA224 0x00000005UL
```

**20.171.1.207 CKG\_MGF1\_SHA256**

```
#define CKG_MGF1_SHA256 0x00000002UL
```

**20.171.1.208 CKG\_MGF1\_SHA384**

```
#define CKG_MGF1_SHA384 0x00000003UL
```

**20.171.1.209 CKG\_MGF1\_SHA512**

```
#define CKG_MGF1_SHA512 0x00000004UL
```

**20.171.1.210 CKH\_CLOCK**

```
#define CKH_CLOCK 0x00000002UL
```

### 20.171.1.211 CKH\_MONOTONIC\_COUNTER

```
#define CKH_MONOTONIC_COUNTER 0x00000001UL
```

### 20.171.1.212 CKH\_USER\_INTERFACE

```
#define CKH_USER_INTERFACE 0x00000003UL
```

### 20.171.1.213 CKH\_VENDOR\_DEFINED

```
#define CKH_VENDOR_DEFINED 0x80000000UL
```

### 20.171.1.214 CKK\_ACTI

```
#define CKK_ACTI 0x00000024UL
```

### 20.171.1.215 CKK\_AES

```
#define CKK_AES 0x0000001FUL
```

### 20.171.1.216 CKK\_ARIA

```
#define CKK_ARIA 0x00000026UL
```

### 20.171.1.217 CKK\_BATON

```
#define CKK_BATON 0x0000001CUL
```

### 20.171.1.218 CKK\_BLOWFISH

```
#define CKK_BLOWFISH 0x00000020UL
```

**20.171.1.219 CKK\_CAMELLIA**

```
#define CKK_CAMELLIA 0x00000025UL
```

**20.171.1.220 CKK\_CAST**

```
#define CKK_CAST 0x00000016UL
```

**20.171.1.221 CKK\_CAST128**

```
#define CKK_CAST128 0x00000018UL
```

**20.171.1.222 CKK\_CAST3**

```
#define CKK_CAST3 0x00000017UL
```

**20.171.1.223 CKK\_CAST5**

```
#define CKK_CAST5 0x00000018UL /* Deprecated */
```

**20.171.1.224 CKK\_CDMF**

```
#define CKK_CDMF 0x0000001EUL
```

**20.171.1.225 CKK\_DES**

```
#define CKK_DES 0x00000013UL
```

**20.171.1.226 CKK\_DES2**

```
#define CKK_DES2 0x00000014UL
```

### 20.171.1.227 CKK\_DES3

```
#define CKK_DES3 0x00000015UL
```

### 20.171.1.228 CKK\_DH

```
#define CKK_DH 0x00000002UL
```

### 20.171.1.229 CKK\_DSA

```
#define CKK_DSA 0x00000001UL
```

### 20.171.1.230 CKK\_EC

```
#define CKK_EC 0x00000003UL
```

### 20.171.1.231 CKK\_ECDSA

```
#define CKK_ECDSA 0x00000003UL /* Deprecated */
```

### 20.171.1.232 CKK\_GENERIC\_SECRET

```
#define CKK_GENERIC_SECRET 0x00000010UL
```

### 20.171.1.233 CKK\_GOST28147

```
#define CKK_GOST28147 0x00000032UL
```

### 20.171.1.234 CKK\_GOSTR3410

```
#define CKK_GOSTR3410 0x00000030UL
```

**20.171.1.235 CKK\_GOSTR3411**

```
#define CKK_GOSTR3411 0x00000031UL
```

**20.171.1.236 CKK\_HOTP**

```
#define CKK_HOTP 0x00000023UL
```

**20.171.1.237 CKK\_IDEA**

```
#define CKK_IDEA 0x0000001AUL
```

**20.171.1.238 CKK\_JUNIPER**

```
#define CKK_JUNIPER 0x0000001DUL
```

**20.171.1.239 CKK\_KEA**

```
#define CKK_KEA 0x00000005UL
```

**20.171.1.240 CKK\_MD5\_HMAC**

```
#define CKK_MD5_HMAC 0x00000027UL
```

**20.171.1.241 CKK\_RC2**

```
#define CKK_RC2 0x00000011UL
```

**20.171.1.242 CKK\_RC4**

```
#define CKK_RC4 0x00000012UL
```

### 20.171.1.243 CKK\_RC5

```
#define CKK_RC5 0x00000019UL
```

### 20.171.1.244 CKK\_RIPEMD128\_HMAC

```
#define CKK_RIPEMD128_HMAC 0x00000029UL
```

### 20.171.1.245 CKK\_RIPEMD160\_HMAC

```
#define CKK_RIPEMD160_HMAC 0x0000002AUL
```

### 20.171.1.246 CKK\_RSA

```
#define CKK_RSA 0x00000000UL
```

### 20.171.1.247 CKK\_SECURID

```
#define CKK_SECURID 0x00000022UL
```

### 20.171.1.248 CKK\_SEED

```
#define CKK_SEED 0x0000002FUL
```

### 20.171.1.249 CKK\_SHA224\_HMAC

```
#define CKK_SHA224_HMAC 0x0000002EUL
```

### 20.171.1.250 CKK\_SHA256\_HMAC

```
#define CKK_SHA256_HMAC 0x0000002BUL
```

**20.171.1.251 CKK\_SHA384\_HMAC**

```
#define CKK_SHA384_HMAC 0x0000002CUL
```

**20.171.1.252 CKK\_SHA512\_HMAC**

```
#define CKK_SHA512_HMAC 0x0000002DUL
```

**20.171.1.253 CKK\_SHA\_1\_HMAC**

```
#define CKK_SHA_1_HMAC 0x00000028UL
```

**20.171.1.254 CKK\_SKIPJACK**

```
#define CKK_SKIPJACK 0x0000001BUL
```

**20.171.1.255 CKK\_TWOFISH**

```
#define CKK_TWOFISH 0x00000021UL
```

**20.171.1.256 CKK\_VENDOR\_DEFINED**

```
#define CKK_VENDOR_DEFINED 0x80000000UL
```

**20.171.1.257 CKK\_X9\_42\_DH**

```
#define CKK_X9_42_DH 0x00000004UL
```

**20.171.1.258 CKM\_ACTI**

```
#define CKM_ACTI 0x000002A0UL
```

### 20.171.1.259 CKM\_ACTI\_KEY\_GEN

```
#define CKM_ACTI_KEY_GEN 0x000002A1UL
```

### 20.171.1.260 CKM\_AES\_CBC

```
#define CKM_AES_CBC 0x00001082UL
```

### 20.171.1.261 CKM\_AES\_CBC\_ENCRYPT\_DATA

```
#define CKM_AES_CBC_ENCRYPT_DATA 0x00001105UL
```

### 20.171.1.262 CKM\_AES\_CBC\_PAD

```
#define CKM_AES_CBC_PAD 0x00001085UL
```

### 20.171.1.263 CKM\_AES\_CCM

```
#define CKM_AES_CCM 0x00001088UL
```

### 20.171.1.264 CKM\_AES\_CFB1

```
#define CKM_AES_CFB1 0x00002108UL
```

### 20.171.1.265 CKM\_AES\_CFB128

```
#define CKM_AES_CFB128 0x00002107UL
```

### 20.171.1.266 CKM\_AES\_CFB64

```
#define CKM_AES_CFB64 0x00002105UL
```



**20.171.1.267 CKM\_AES\_CFB8**

```
#define CKM_AES_CFB8 0x00002106UL
```

**20.171.1.268 CKM\_AES\_CMAC**

```
#define CKM_AES_CMAC 0x0000108AUL
```

**20.171.1.269 CKM\_AES\_CMAC\_GENERAL**

```
#define CKM_AES_CMAC_GENERAL 0x0000108BUL
```

**20.171.1.270 CKM\_AES\_CTR**

```
#define CKM_AES_CTR 0x00001086UL
```

**20.171.1.271 CKM\_AES\_CTS**

```
#define CKM_AES_CTS 0x00001089UL
```

**20.171.1.272 CKM\_AES\_ECB**

```
#define CKM_AES_ECB 0x00001081UL
```

**20.171.1.273 CKM\_AES\_ECB\_ENCRYPT\_DATA**

```
#define CKM_AES_ECB_ENCRYPT_DATA 0x00001104UL
```

**20.171.1.274 CKM\_AES\_GCM**

```
#define CKM_AES_GCM 0x00001087UL
```

### 20.171.1.275 CKM\_AES\_GMAC

```
#define CKM_AES_GMAC 0x0000108EUL
```

### 20.171.1.276 CKM\_AES\_KEY\_GEN

```
#define CKM_AES_KEY_GEN 0x00001080UL
```

### 20.171.1.277 CKM\_AES\_KEY\_WRAP

```
#define CKM_AES_KEY_WRAP 0x00002109UL /* WAS: 0x00001090 */
```

### 20.171.1.278 CKM\_AES\_KEY\_WRAP\_PAD

```
#define CKM_AES_KEY_WRAP_PAD 0x0000210AUL /* WAS: 0x00001091 */
```

### 20.171.1.279 CKM\_AES\_MAC

```
#define CKM_AES_MAC 0x00001083UL
```

### 20.171.1.280 CKM\_AES\_MAC\_GENERAL

```
#define CKM_AES_MAC_GENERAL 0x00001084UL
```

### 20.171.1.281 CKM\_AES\_OFB

```
#define CKM_AES_OFB 0x00002104UL
```

### 20.171.1.282 CKM\_AES\_XCBC\_MAC

```
#define CKM_AES_XCBC_MAC 0x0000108CUL
```

**20.171.1.283 CKM\_AES\_XCBC\_MAC\_96**

```
#define CKM_AES_XCBC_MAC_96 0x0000108DUL
```

**20.171.1.284 CKM\_ARIA\_CBC**

```
#define CKM_ARIA_CBC 0x00000562UL
```

**20.171.1.285 CKM\_ARIA\_CBC\_ENCRYPT\_DATA**

```
#define CKM_ARIA_CBC_ENCRYPT_DATA 0x00000567UL
```

**20.171.1.286 CKM\_ARIA\_CBC\_PAD**

```
#define CKM_ARIA_CBC_PAD 0x00000565UL
```

**20.171.1.287 CKM\_ARIA\_ECB**

```
#define CKM_ARIA_ECB 0x00000561UL
```

**20.171.1.288 CKM\_ARIA\_ECB\_ENCRYPT\_DATA**

```
#define CKM_ARIA_ECB_ENCRYPT_DATA 0x00000566UL
```

**20.171.1.289 CKM\_ARIA\_KEY\_GEN**

```
#define CKM_ARIA_KEY_GEN 0x00000560UL
```

**20.171.1.290 CKM\_ARIA\_MAC**

```
#define CKM_ARIA_MAC 0x00000563UL
```

### 20.171.1.291 CKM\_ARIA\_MAC\_GENERAL

```
#define CKM_ARIA_MAC_GENERAL 0x00000564UL
```

### 20.171.1.292 CKM\_BATON\_CBC128

```
#define CKM_BATON_CBC128 0x00001033UL
```

### 20.171.1.293 CKM\_BATON\_COUNTER

```
#define CKM_BATON_COUNTER 0x00001034UL
```

### 20.171.1.294 CKM\_BATON\_ECB128

```
#define CKM_BATON_ECB128 0x00001031UL
```

### 20.171.1.295 CKM\_BATON\_ECB96

```
#define CKM_BATON_ECB96 0x00001032UL
```

### 20.171.1.296 CKM\_BATON\_KEY\_GEN

```
#define CKM_BATON_KEY_GEN 0x00001030UL
```

### 20.171.1.297 CKM\_BATON\_SHUFFLE

```
#define CKM_BATON_SHUFFLE 0x00001035UL
```

### 20.171.1.298 CKM\_BATON\_WRAP

```
#define CKM_BATON_WRAP 0x00001036UL
```

**20.171.1.299 CKM\_BLOWFISH\_CBC**

```
#define CKM_BLOWFISH_CBC 0x00001091UL
```

**20.171.1.300 CKM\_BLOWFISH\_CBC\_PAD**

```
#define CKM_BLOWFISH_CBC_PAD 0x00001094UL
```

**20.171.1.301 CKM\_BLOWFISH\_KEY\_GEN**

```
#define CKM_BLOWFISH_KEY_GEN 0x00001090UL
```

**20.171.1.302 CKM\_CAMELLIA\_CBC**

```
#define CKM_CAMELLIA_CBC 0x00000552UL
```

**20.171.1.303 CKM\_CAMELLIA\_CBC\_ENCRYPT\_DATA**

```
#define CKM_CAMELLIA_CBC_ENCRYPT_DATA 0x00000557UL
```

**20.171.1.304 CKM\_CAMELLIA\_CBC\_PAD**

```
#define CKM_CAMELLIA_CBC_PAD 0x00000555UL
```

**20.171.1.305 CKM\_CAMELLIA\_CTR**

```
#define CKM_CAMELLIA_CTR 0x00000558UL
```

**20.171.1.306 CKM\_CAMELLIA\_ECB**

```
#define CKM_CAMELLIA_ECB 0x00000551UL
```

### 20.171.1.307 CKM\_CAMELLIA\_ECB\_ENCRYPT\_DATA

```
#define CKM_CAMELLIA_ECB_ENCRYPT_DATA 0x00000556UL
```

### 20.171.1.308 CKM\_CAMELLIA\_KEY\_GEN

```
#define CKM_CAMELLIA_KEY_GEN 0x00000550UL
```

### 20.171.1.309 CKM\_CAMELLIA\_MAC

```
#define CKM_CAMELLIA_MAC 0x00000553UL
```

### 20.171.1.310 CKM\_CAMELLIA\_MAC\_GENERAL

```
#define CKM_CAMELLIA_MAC_GENERAL 0x00000554UL
```

### 20.171.1.311 CKM\_CAST128\_CBC

```
#define CKM_CAST128_CBC 0x00000322UL
```

### 20.171.1.312 CKM\_CAST128\_CBC\_PAD

```
#define CKM_CAST128_CBC_PAD 0x00000325UL
```

### 20.171.1.313 CKM\_CAST128\_ECB

```
#define CKM_CAST128_ECB 0x00000321UL
```

### 20.171.1.314 CKM\_CAST128\_KEY\_GEN

```
#define CKM_CAST128_KEY_GEN 0x00000320UL
```

**20.171.1.315 CKM\_CAST128\_MAC**

```
#define CKM_CAST128_MAC 0x00000323UL
```

**20.171.1.316 CKM\_CAST128\_MAC\_GENERAL**

```
#define CKM_CAST128_MAC_GENERAL 0x00000324UL
```

**20.171.1.317 CKM\_CAST3\_CBC**

```
#define CKM_CAST3_CBC 0x00000312UL
```

**20.171.1.318 CKM\_CAST3\_CBC\_PAD**

```
#define CKM_CAST3_CBC_PAD 0x00000315UL
```

**20.171.1.319 CKM\_CAST3\_ECB**

```
#define CKM_CAST3_ECB 0x00000311UL
```

**20.171.1.320 CKM\_CAST3\_KEY\_GEN**

```
#define CKM_CAST3_KEY_GEN 0x00000310UL
```

**20.171.1.321 CKM\_CAST3\_MAC**

```
#define CKM_CAST3_MAC 0x00000313UL
```

**20.171.1.322 CKM\_CAST3\_MAC\_GENERAL**

```
#define CKM_CAST3_MAC_GENERAL 0x00000314UL
```

### 20.171.1.323 CKM\_CAST5\_CBC

```
#define CKM_CAST5_CBC 0x00000322UL /* Deprecated */
```

### 20.171.1.324 CKM\_CAST5\_CBC\_PAD

```
#define CKM_CAST5_CBC_PAD 0x00000325UL /* Deprecated */
```

### 20.171.1.325 CKM\_CAST5\_ECB

```
#define CKM_CAST5_ECB 0x00000321UL
```

### 20.171.1.326 CKM\_CAST5\_KEY\_GEN

```
#define CKM_CAST5_KEY_GEN 0x00000320UL
```

### 20.171.1.327 CKM\_CAST5\_MAC

```
#define CKM_CAST5_MAC 0x00000323UL /* Deprecated */
```

### 20.171.1.328 CKM\_CAST5\_MAC\_GENERAL

```
#define CKM_CAST5_MAC_GENERAL 0x00000324UL /* Deprecated */
```

### 20.171.1.329 CKM\_CAST\_CBC

```
#define CKM_CAST_CBC 0x00000302UL
```

### 20.171.1.330 CKM\_CAST\_CBC\_PAD

```
#define CKM_CAST_CBC_PAD 0x00000305UL
```



**20.171.1.331 CKM\_CAST\_ECB**

```
#define CKM_CAST_ECB 0x00000301UL
```

**20.171.1.332 CKM\_CAST\_KEY\_GEN**

```
#define CKM_CAST_KEY_GEN 0x00000300UL
```

**20.171.1.333 CKM\_CAST\_MAC**

```
#define CKM_CAST_MAC 0x00000303UL
```

**20.171.1.334 CKM\_CAST\_MAC\_GENERAL**

```
#define CKM_CAST_MAC_GENERAL 0x00000304UL
```

**20.171.1.335 CKM\_CDMF\_CBC**

```
#define CKM_CDMF_CBC 0x00000142UL
```

**20.171.1.336 CKM\_CDMF\_CBC\_PAD**

```
#define CKM_CDMF_CBC_PAD 0x00000145UL
```

**20.171.1.337 CKM\_CDMF\_ECB**

```
#define CKM_CDMF_ECB 0x00000141UL
```

**20.171.1.338 CKM\_CDMF\_KEY\_GEN**

```
#define CKM_CDMF_KEY_GEN 0x00000140UL
```

### 20.171.1.339 CKM\_CDMF\_MAC

```
#define CKM_CDMF_MAC 0x00000143UL
```

### 20.171.1.340 CKM\_CDMF\_MAC\_GENERAL

```
#define CKM_CDMF_MAC_GENERAL 0x00000144UL
```

### 20.171.1.341 CKM\_CMS\_SIG

```
#define CKM_CMS_SIG 0x00000500UL
```

### 20.171.1.342 CKM\_CONCATENATE\_BASE\_AND\_DATA

```
#define CKM_CONCATENATE_BASE_AND_DATA 0x00000362UL
```

### 20.171.1.343 CKM\_CONCATENATE\_BASE\_AND\_KEY

```
#define CKM_CONCATENATE_BASE_AND_KEY 0x00000360UL
```

### 20.171.1.344 CKM\_CONCATENATE\_DATA\_AND\_BASE

```
#define CKM_CONCATENATE_DATA_AND_BASE 0x00000363UL
```

### 20.171.1.345 CKM\_DES2\_KEY\_GEN

```
#define CKM_DES2_KEY_GEN 0x00000130UL
```

### 20.171.1.346 CKM\_DES3\_CBC

```
#define CKM_DES3_CBC 0x00000133UL
```

**20.171.1.347 CKM\_DES3\_CBC\_ENCRYPT\_DATA**

```
#define CKM_DES3_CBC_ENCRYPT_DATA 0x00001103UL
```

**20.171.1.348 CKM\_DES3\_CBC\_PAD**

```
#define CKM_DES3_CBC_PAD 0x00000136UL
```

**20.171.1.349 CKM\_DES3\_CMAC**

```
#define CKM_DES3_CMAC 0x00000138UL
```

**20.171.1.350 CKM\_DES3\_CMAC\_GENERAL**

```
#define CKM_DES3_CMAC_GENERAL 0x00000137UL
```

**20.171.1.351 CKM\_DES3\_ECB**

```
#define CKM_DES3_ECB 0x00000132UL
```

**20.171.1.352 CKM\_DES3\_ECB\_ENCRYPT\_DATA**

```
#define CKM_DES3_ECB_ENCRYPT_DATA 0x00001102UL
```

**20.171.1.353 CKM\_DES3\_KEY\_GEN**

```
#define CKM_DES3_KEY_GEN 0x00000131UL
```

**20.171.1.354 CKM\_DES3\_MAC**

```
#define CKM_DES3_MAC 0x00000134UL
```

### 20.171.1.355 CKM\_DES3\_MAC\_GENERAL

```
#define CKM_DES3_MAC_GENERAL 0x00000135UL
```

### 20.171.1.356 CKM\_DES\_CBC

```
#define CKM_DES_CBC 0x00000122UL
```

### 20.171.1.357 CKM\_DES\_CBC\_ENCRYPT\_DATA

```
#define CKM_DES_CBC_ENCRYPT_DATA 0x00001101UL
```

### 20.171.1.358 CKM\_DES\_CBC\_PAD

```
#define CKM_DES_CBC_PAD 0x00000125UL
```

### 20.171.1.359 CKM\_DES\_CFB64

```
#define CKM_DES_CFB64 0x00000152UL
```

### 20.171.1.360 CKM\_DES\_CFB8

```
#define CKM_DES_CFB8 0x00000153UL
```

### 20.171.1.361 CKM\_DES\_ECB

```
#define CKM_DES_ECB 0x00000121UL
```

### 20.171.1.362 CKM\_DES\_ECB\_ENCRYPT\_DATA

```
#define CKM_DES_ECB_ENCRYPT_DATA 0x00001100UL
```

**20.171.1.363 CKM\_DES\_KEY\_GEN**

```
#define CKM_DES_KEY_GEN 0x00000120UL
```

**20.171.1.364 CKM\_DES\_MAC**

```
#define CKM_DES_MAC 0x00000123UL
```

**20.171.1.365 CKM\_DES\_MAC\_GENERAL**

```
#define CKM_DES_MAC_GENERAL 0x00000124UL
```

**20.171.1.366 CKM\_DES\_OFB64**

```
#define CKM_DES_OFB64 0x00000150UL
```

**20.171.1.367 CKM\_DES\_OFB8**

```
#define CKM_DES_OFB8 0x00000151UL
```

**20.171.1.368 CKM\_DH\_PKCS\_DERIVE**

```
#define CKM_DH_PKCS_DERIVE 0x00000021UL
```

**20.171.1.369 CKM\_DH\_PKCS\_KEY\_PAIR\_GEN**

```
#define CKM_DH_PKCS_KEY_PAIR_GEN 0x00000020UL
```

**20.171.1.370 CKM\_DH\_PKCS\_PARAMETER\_GEN**

```
#define CKM_DH_PKCS_PARAMETER_GEN 0x00002001UL
```

### 20.171.1.371 CKM\_DSA

```
#define CKM_DSA 0x00000011UL
```

### 20.171.1.372 CKM\_DSA\_KEY\_PAIR\_GEN

```
#define CKM_DSA_KEY_PAIR_GEN 0x00000010UL
```

### 20.171.1.373 CKM\_DSA\_PARAMETER\_GEN

```
#define CKM_DSA_PARAMETER_GEN 0x00002000UL
```

### 20.171.1.374 CKM\_DSA\_PROBABLISTIC\_PARAMETER\_GEN

```
#define CKM_DSA_PROBABLISTIC_PARAMETER_GEN 0x00002003UL
```

### 20.171.1.375 CKM\_DSA\_SHA1

```
#define CKM_DSA_SHA1 0x00000012UL
```

### 20.171.1.376 CKM\_DSA\_SHA224

```
#define CKM_DSA_SHA224 0x00000013UL
```

### 20.171.1.377 CKM\_DSA\_SHA256

```
#define CKM_DSA_SHA256 0x00000014UL
```

### 20.171.1.378 CKM\_DSA\_SHA384

```
#define CKM_DSA_SHA384 0x00000015UL
```

**20.171.1.379 CKM\_DSA\_SHA512**

```
#define CKM_DSA_SHA512 0x00000016UL
```

**20.171.1.380 CKM\_DSA\_SHAWA\_TAYLOR\_PARAMETER\_GEN**

```
#define CKM_DSA_SHAWA_TAYLOR_PARAMETER_GEN 0x00002004UL
```

**20.171.1.381 CKM\_EC\_KEY\_PAIR\_GEN**

```
#define CKM_EC_KEY_PAIR_GEN 0x00001040UL
```

**20.171.1.382 CKM\_ECDH1\_COFACTOR\_DERIVE**

```
#define CKM_ECDH1_COFACTOR_DERIVE 0x00001051UL
```

**20.171.1.383 CKM\_ECDH1\_DERIVE**

```
#define CKM_ECDH1_DERIVE 0x00001050UL
```

**20.171.1.384 CKM\_ECDH\_AES\_KEY\_WRAP**

```
#define CKM_ECDH_AES_KEY_WRAP 0x00001053UL
```

**20.171.1.385 CKM\_ECDSA**

```
#define CKM_ECDSA 0x00001041UL
```

**20.171.1.386 CKM\_ECDSA\_KEY\_PAIR\_GEN**

```
#define CKM_ECDSA_KEY_PAIR_GEN 0x00001040UL /* Deprecated */
```

### 20.171.1.387 CKM\_ECDSA\_SHA1

```
#define CKM_ECDSA_SHA1 0x00001042UL
```

### 20.171.1.388 CKM\_ECDSA\_SHA224

```
#define CKM_ECDSA_SHA224 0x00001043UL
```

### 20.171.1.389 CKM\_ECDSA\_SHA256

```
#define CKM_ECDSA_SHA256 0x00001044UL
```

### 20.171.1.390 CKM\_ECDSA\_SHA384

```
#define CKM_ECDSA_SHA384 0x00001045UL
```

### 20.171.1.391 CKM\_ECDSA\_SHA512

```
#define CKM_ECDSA_SHA512 0x00001046UL
```

### 20.171.1.392 CKM\_ECMQV\_DERIVE

```
#define CKM_ECMQV_DERIVE 0x00001052UL
```

### 20.171.1.393 CKM\_EXTRACT\_KEY\_FROM\_KEY

```
#define CKM_EXTRACT_KEY_FROM_KEY 0x00000365UL
```

### 20.171.1.394 CKM\_FASTHASH

```
#define CKM_FASTHASH 0x00001070UL
```



**20.171.1.395 CKM\_FORTEZZA\_TIMESTAMP**

```
#define CKM_FORTEZZA_TIMESTAMP 0x00001020UL
```

**20.171.1.396 CKM\_GENERIC\_SECRET\_KEY\_GEN**

```
#define CKM_GENERIC_SECRET_KEY_GEN 0x00000350UL
```

**20.171.1.397 CKM\_GOST28147**

```
#define CKM_GOST28147 0x00001222UL
```

**20.171.1.398 CKM\_GOST28147\_ECB**

```
#define CKM_GOST28147_ECB 0x00001221UL
```

**20.171.1.399 CKM\_GOST28147\_KEY\_GEN**

```
#define CKM_GOST28147_KEY_GEN 0x00001220UL
```

**20.171.1.400 CKM\_GOST28147\_KEY\_WRAP**

```
#define CKM_GOST28147_KEY_WRAP 0x00001224UL
```

**20.171.1.401 CKM\_GOST28147\_MAC**

```
#define CKM_GOST28147_MAC 0x00001223UL
```

**20.171.1.402 CKM\_GOSTR3410**

```
#define CKM_GOSTR3410 0x00001201UL
```

### 20.171.1.403 CKM\_GOSTR3410\_DERIVE

```
#define CKM_GOSTR3410_DERIVE 0x00001204UL
```

### 20.171.1.404 CKM\_GOSTR3410\_KEY\_PAIR\_GEN

```
#define CKM_GOSTR3410_KEY_PAIR_GEN 0x00001200UL
```

### 20.171.1.405 CKM\_GOSTR3410\_KEY\_WRAP

```
#define CKM_GOSTR3410_KEY_WRAP 0x00001203UL
```

### 20.171.1.406 CKM\_GOSTR3410\_WITH\_GOSTR3411

```
#define CKM_GOSTR3410_WITH_GOSTR3411 0x00001202UL
```

### 20.171.1.407 CKM\_GOSTR3411

```
#define CKM_GOSTR3411 0x00001210UL
```

### 20.171.1.408 CKM\_GOSTR3411\_HMAC

```
#define CKM_GOSTR3411_HMAC 0x00001211UL
```

### 20.171.1.409 CKM\_HOTP

```
#define CKM_HOTP 0x00000291UL
```

### 20.171.1.410 CKM\_HOTP\_KEY\_GEN

```
#define CKM_HOTP_KEY_GEN 0x00000290UL
```

**20.171.1.411 CKM\_IDEA\_CBC**

```
#define CKM_IDEA_CBC 0x00000342UL
```

**20.171.1.412 CKM\_IDEA\_CBC\_PAD**

```
#define CKM_IDEA_CBC_PAD 0x00000345UL
```

**20.171.1.413 CKM\_IDEA\_ECB**

```
#define CKM_IDEA_ECB 0x00000341UL
```

**20.171.1.414 CKM\_IDEA\_KEY\_GEN**

```
#define CKM_IDEA_KEY_GEN 0x00000340UL
```

**20.171.1.415 CKM\_IDEA\_MAC**

```
#define CKM_IDEA_MAC 0x00000343UL
```

**20.171.1.416 CKM\_IDEA\_MAC\_GENERAL**

```
#define CKM_IDEA_MAC_GENERAL 0x00000344UL
```

**20.171.1.417 CKM\_JUNIPER\_CBC128**

```
#define CKM_JUNIPER_CBC128 0x00001062UL
```

**20.171.1.418 CKM\_JUNIPER\_COUNTER**

```
#define CKM_JUNIPER_COUNTER 0x00001063UL
```

### 20.171.1.419 CKM\_JUNIPER\_ECB128

```
#define CKM_JUNIPER_ECB128 0x00001061UL
```

### 20.171.1.420 CKM\_JUNIPER\_KEY\_GEN

```
#define CKM_JUNIPER_KEY_GEN 0x00001060UL
```

### 20.171.1.421 CKM\_JUNIPER\_SHUFFLE

```
#define CKM_JUNIPER_SHUFFLE 0x00001064UL
```

### 20.171.1.422 CKM\_JUNIPER\_WRAP

```
#define CKM_JUNIPER_WRAP 0x00001065UL
```

### 20.171.1.423 CKM\_KEA\_DERIVE

```
#define CKM_KEA_DERIVE 0x00001012UL
```

### 20.171.1.424 CKM\_KEA\_KEY\_DERIVE

```
#define CKM_KEA_KEY_DERIVE 0x00001011UL
```

### 20.171.1.425 CKM\_KEA\_KEY\_PAIR\_GEN

```
#define CKM_KEA_KEY_PAIR_GEN 0x00001010UL
```

### 20.171.1.426 CKM\_KEY\_WRAP\_LYNKS

```
#define CKM_KEY_WRAP_LYNKS 0x00000400UL
```

**20.171.1.427 CKM\_KEY\_WRAP\_SET\_OAEP**

```
#define CKM_KEY_WRAP_SET_OAEP 0x00000401UL
```

**20.171.1.428 CKM\_KIP\_DERIVE**

```
#define CKM_KIP_DERIVE 0x00000510UL
```

**20.171.1.429 CKM\_KIP\_MAC**

```
#define CKM_KIP_MAC 0x00000512UL
```

**20.171.1.430 CKM\_KIP\_WRAP**

```
#define CKM_KIP_WRAP 0x00000511UL
```

**20.171.1.431 CKM\_MD2**

```
#define CKM_MD2 0x00000200UL
```

**20.171.1.432 CKM\_MD2\_HMAC**

```
#define CKM_MD2_HMAC 0x00000201UL
```

**20.171.1.433 CKM\_MD2\_HMAC\_GENERAL**

```
#define CKM_MD2_HMAC_GENERAL 0x00000202UL
```

**20.171.1.434 CKM\_MD2\_KEY\_DERIVATION**

```
#define CKM_MD2_KEY_DERIVATION 0x00000391UL
```

### 20.171.1.435 CKM\_MD2\_RSA\_PKCS

```
#define CKM_MD2_RSA_PKCS 0x00000004UL
```

### 20.171.1.436 CKM\_MD5

```
#define CKM_MD5 0x00000210UL
```

### 20.171.1.437 CKM\_MD5\_HMAC

```
#define CKM_MD5_HMAC 0x00000211UL
```

### 20.171.1.438 CKM\_MD5\_HMAC\_GENERAL

```
#define CKM_MD5_HMAC_GENERAL 0x00000212UL
```

### 20.171.1.439 CKM\_MD5\_KEY\_DERIVATION

```
#define CKM_MD5_KEY_DERIVATION 0x00000390UL
```

### 20.171.1.440 CKM\_MD5\_RSA\_PKCS

```
#define CKM_MD5_RSA_PKCS 0x00000005UL
```

### 20.171.1.441 CKM\_PBA\_SHA1\_WITH\_SHA1\_HMAC

```
#define CKM_PBA_SHA1_WITH_SHA1_HMAC 0x000003C0UL
```

### 20.171.1.442 CKM\_PBE\_MD2\_DES\_CBC

```
#define CKM_PBE_MD2_DES_CBC 0x000003A0UL
```

**20.171.1.443 CKM\_PBE\_MD5\_CAST128\_CBC**

```
#define CKM_PBE_MD5_CAST128_CBC 0x000003A4UL
```

**20.171.1.444 CKM\_PBE\_MD5\_CAST3\_CBC**

```
#define CKM_PBE_MD5_CAST3_CBC 0x000003A3UL
```

**20.171.1.445 CKM\_PBE\_MD5\_CAST5\_CBC**

```
#define CKM_PBE_MD5_CAST5_CBC 0x000003A4UL /* Deprecated */
```

**20.171.1.446 CKM\_PBE\_MD5\_CAST\_CBC**

```
#define CKM_PBE_MD5_CAST_CBC 0x000003A2UL
```

**20.171.1.447 CKM\_PBE\_MD5\_DES\_CBC**

```
#define CKM_PBE_MD5_DES_CBC 0x000003A1UL
```

**20.171.1.448 CKM\_PBE\_SHA1\_CAST128\_CBC**

```
#define CKM_PBE_SHA1_CAST128_CBC 0x000003A5UL
```

**20.171.1.449 CKM\_PBE\_SHA1\_CAST5\_CBC**

```
#define CKM_PBE_SHA1_CAST5_CBC 0x000003A5UL /* Deprecated */
```

**20.171.1.450 CKM\_PBE\_SHA1\_DES2\_EDE\_CBC**

```
#define CKM_PBE_SHA1_DES2_EDE_CBC 0x000003A9UL
```

### 20.171.1.451 CKM\_PBE\_SHA1\_DES3\_EDE\_CBC

```
#define CKM_PBE_SHA1_DES3_EDE_CBC 0x000003A8UL
```

### 20.171.1.452 CKM\_PBE\_SHA1\_RC2\_128\_CBC

```
#define CKM_PBE_SHA1_RC2_128_CBC 0x000003AAUL
```

### 20.171.1.453 CKM\_PBE\_SHA1\_RC2\_40\_CBC

```
#define CKM_PBE_SHA1_RC2_40_CBC 0x000003ABUL
```

### 20.171.1.454 CKM\_PBE\_SHA1\_RC4\_128

```
#define CKM_PBE_SHA1_RC4_128 0x000003A6UL
```

### 20.171.1.455 CKM\_PBE\_SHA1\_RC4\_40

```
#define CKM_PBE_SHA1_RC4_40 0x000003A7UL
```

### 20.171.1.456 CKM\_PKCS5\_PBKD2

```
#define CKM_PKCS5_PBKD2 0x000003B0UL
```

### 20.171.1.457 CKM\_RC2\_CBC

```
#define CKM_RC2_CBC 0x00000102UL
```

### 20.171.1.458 CKM\_RC2\_CBC\_PAD

```
#define CKM_RC2_CBC_PAD 0x00000105UL
```



**20.171.1.459 CKM\_RC2\_ECB**

```
#define CKM_RC2_ECB 0x00000101UL
```

**20.171.1.460 CKM\_RC2\_KEY\_GEN**

```
#define CKM_RC2_KEY_GEN 0x00000100UL
```

**20.171.1.461 CKM\_RC2\_MAC**

```
#define CKM_RC2_MAC 0x00000103UL
```

**20.171.1.462 CKM\_RC2\_MAC\_GENERAL**

```
#define CKM_RC2_MAC_GENERAL 0x00000104UL
```

**20.171.1.463 CKM\_RC4**

```
#define CKM_RC4 0x00000111UL
```

**20.171.1.464 CKM\_RC4\_KEY\_GEN**

```
#define CKM_RC4_KEY_GEN 0x00000110UL
```

**20.171.1.465 CKM\_RC5\_CBC**

```
#define CKM_RC5_CBC 0x00000332UL
```

**20.171.1.466 CKM\_RC5\_CBC\_PAD**

```
#define CKM_RC5_CBC_PAD 0x00000335UL
```

### 20.171.1.467 CKM\_RC5\_ECB

```
#define CKM_RC5_ECB 0x00000331UL
```

### 20.171.1.468 CKM\_RC5\_KEY\_GEN

```
#define CKM_RC5_KEY_GEN 0x00000330UL
```

### 20.171.1.469 CKM\_RC5\_MAC

```
#define CKM_RC5_MAC 0x00000333UL
```

### 20.171.1.470 CKM\_RC5\_MAC\_GENERAL

```
#define CKM_RC5_MAC_GENERAL 0x00000334UL
```

### 20.171.1.471 CKM\_RIPEMD128

```
#define CKM_RIPEMD128 0x00000230UL
```

### 20.171.1.472 CKM\_RIPEMD128\_HMAC

```
#define CKM_RIPEMD128_HMAC 0x00000231UL
```

### 20.171.1.473 CKM\_RIPEMD128\_HMAC\_GENERAL

```
#define CKM_RIPEMD128_HMAC_GENERAL 0x00000232UL
```

### 20.171.1.474 CKM\_RIPEMD128\_RSA\_PKCS

```
#define CKM_RIPEMD128_RSA_PKCS 0x00000007UL
```

**20.171.1.475 CKM\_RIPEMD160**

```
#define CKM_RIPEMD160 0x00000240UL
```

**20.171.1.476 CKM\_RIPEMD160\_HMAC**

```
#define CKM_RIPEMD160_HMAC 0x00000241UL
```

**20.171.1.477 CKM\_RIPEMD160\_HMAC\_GENERAL**

```
#define CKM_RIPEMD160_HMAC_GENERAL 0x00000242UL
```

**20.171.1.478 CKM\_RIPEMD160\_RSA\_PKCS**

```
#define CKM_RIPEMD160_RSA_PKCS 0x00000008UL
```

**20.171.1.479 CKM\_RSA\_9796**

```
#define CKM_RSA_9796 0x00000002UL
```

**20.171.1.480 CKM\_RSA\_AES\_KEY\_WRAP**

```
#define CKM_RSA_AES_KEY_WRAP 0x00001054UL
```

**20.171.1.481 CKM\_RSA\_PKCS**

```
#define CKM_RSA_PKCS 0x00000001UL
```

**20.171.1.482 CKM\_RSA\_PKCS\_KEY\_PAIR\_GEN**

```
#define CKM_RSA_PKCS_KEY_PAIR_GEN 0x00000000UL
```

### 20.171.1.483 CKM\_RSA\_PKCS\_OAEP

```
#define CKM_RSA_PKCS_OAEP 0x00000009UL
```

### 20.171.1.484 CKM\_RSA\_PKCS\_OAEP\_TPM\_1\_1

```
#define CKM_RSA_PKCS_OAEP_TPM_1_1 0x00004002UL
```

### 20.171.1.485 CKM\_RSA\_PKCS\_PSS

```
#define CKM_RSA_PKCS_PSS 0x0000000DUL
```

### 20.171.1.486 CKM\_RSA\_PKCS\_TPM\_1\_1

```
#define CKM_RSA_PKCS_TPM_1_1 0x00004001UL
```

### 20.171.1.487 CKM\_RSA\_X9\_31

```
#define CKM_RSA_X9_31 0x0000000BUL
```

### 20.171.1.488 CKM\_RSA\_X9\_31\_KEY\_PAIR\_GEN

```
#define CKM_RSA_X9_31_KEY_PAIR_GEN 0x0000000AUL
```

### 20.171.1.489 CKM\_RSA\_X\_509

```
#define CKM_RSA_X_509 0x00000003UL
```

### 20.171.1.490 CKM\_SECURID

```
#define CKM_SECURID 0x00000282UL
```

**20.171.1.491 CKM\_SECURID\_KEY\_GEN**

```
#define CKM_SECURID_KEY_GEN 0x00000280UL
```

**20.171.1.492 CKM\_SEED\_CBC**

```
#define CKM_SEED_CBC 0x00000652UL
```

**20.171.1.493 CKM\_SEED\_CBC\_ENCRYPT\_DATA**

```
#define CKM_SEED_CBC_ENCRYPT_DATA 0x00000657UL
```

**20.171.1.494 CKM\_SEED\_CBC\_PAD**

```
#define CKM_SEED_CBC_PAD 0x00000655UL
```

**20.171.1.495 CKM\_SEED\_ECB**

```
#define CKM_SEED_ECB 0x00000651UL
```

**20.171.1.496 CKM\_SEED\_ECB\_ENCRYPT\_DATA**

```
#define CKM_SEED_ECB_ENCRYPT_DATA 0x00000656UL
```

**20.171.1.497 CKM\_SEED\_KEY\_GEN**

```
#define CKM_SEED_KEY_GEN 0x00000650UL
```

**20.171.1.498 CKM\_SEED\_MAC**

```
#define CKM_SEED_MAC 0x00000653UL
```

### 20.171.1.499 CKM\_SEED\_MAC\_GENERAL

```
#define CKM_SEED_MAC_GENERAL 0x00000654UL
```

### 20.171.1.500 CKM\_SHA1\_KEY\_DERIVATION

```
#define CKM_SHA1_KEY_DERIVATION 0x00000392UL
```

### 20.171.1.501 CKM\_SHA1\_RSA\_PKCS

```
#define CKM_SHA1_RSA_PKCS 0x00000006UL
```

### 20.171.1.502 CKM\_SHA1\_RSA\_PKCS\_PSS

```
#define CKM_SHA1_RSA_PKCS_PSS 0x0000000EUL
```

### 20.171.1.503 CKM\_SHA1\_RSA\_X9\_31

```
#define CKM_SHA1_RSA_X9_31 0x0000000CUL
```

### 20.171.1.504 CKM\_SHA224

```
#define CKM_SHA224 0x00000255UL
```

### 20.171.1.505 CKM\_SHA224\_HMAC

```
#define CKM_SHA224_HMAC 0x00000256UL
```

### 20.171.1.506 CKM\_SHA224\_HMAC\_GENERAL

```
#define CKM_SHA224_HMAC_GENERAL 0x00000257UL
```

**20.171.1.507 CKM\_SHA224\_KEY\_DERIVATION**

```
#define CKM_SHA224_KEY_DERIVATION 0x00000396UL
```

**20.171.1.508 CKM\_SHA224\_RSA\_PKCS**

```
#define CKM_SHA224_RSA_PKCS 0x00000046UL
```

**20.171.1.509 CKM\_SHA224\_RSA\_PKCS\_PSS**

```
#define CKM_SHA224_RSA_PKCS_PSS 0x00000047UL
```

**20.171.1.510 CKM\_SHA256**

```
#define CKM_SHA256 0x00000250UL
```

**20.171.1.511 CKM\_SHA256\_HMAC**

```
#define CKM_SHA256_HMAC 0x00000251UL
```

**20.171.1.512 CKM\_SHA256\_HMAC\_GENERAL**

```
#define CKM_SHA256_HMAC_GENERAL 0x00000252UL
```

**20.171.1.513 CKM\_SHA256\_KEY\_DERIVATION**

```
#define CKM_SHA256_KEY_DERIVATION 0x00000393UL
```

**20.171.1.514 CKM\_SHA256\_RSA\_PKCS**

```
#define CKM_SHA256_RSA_PKCS 0x00000040UL
```

### 20.171.1.515 CKM\_SHA256\_RSA\_PKCS\_PSS

```
#define CKM_SHA256_RSA_PKCS_PSS 0x00000043UL
```

### 20.171.1.516 CKM\_SHA384

```
#define CKM_SHA384 0x00000260UL
```

### 20.171.1.517 CKM\_SHA384\_HMAC

```
#define CKM_SHA384_HMAC 0x00000261UL
```

### 20.171.1.518 CKM\_SHA384\_HMAC\_GENERAL

```
#define CKM_SHA384_HMAC_GENERAL 0x00000262UL
```

### 20.171.1.519 CKM\_SHA384\_KEY\_DERIVATION

```
#define CKM_SHA384_KEY_DERIVATION 0x00000394UL
```

### 20.171.1.520 CKM\_SHA384\_RSA\_PKCS

```
#define CKM_SHA384_RSA_PKCS 0x00000041UL
```

### 20.171.1.521 CKM\_SHA384\_RSA\_PKCS\_PSS

```
#define CKM_SHA384_RSA_PKCS_PSS 0x00000044UL
```

### 20.171.1.522 CKM\_SHA512

```
#define CKM_SHA512 0x00000270UL
```



**20.171.1.523 CKM\_SHA512\_224**

```
#define CKM_SHA512_224 0x000000048UL
```

**20.171.1.524 CKM\_SHA512\_224\_HMAC**

```
#define CKM_SHA512_224_HMAC 0x000000049UL
```

**20.171.1.525 CKM\_SHA512\_224\_HMAC\_GENERAL**

```
#define CKM_SHA512_224_HMAC_GENERAL 0x00000004AUL
```

**20.171.1.526 CKM\_SHA512\_224\_KEY\_DERIVATION**

```
#define CKM_SHA512_224_KEY_DERIVATION 0x00000004BUL
```

**20.171.1.527 CKM\_SHA512\_256**

```
#define CKM_SHA512_256 0x00000004CUL
```

**20.171.1.528 CKM\_SHA512\_256\_HMAC**

```
#define CKM_SHA512_256_HMAC 0x00000004DUL
```

**20.171.1.529 CKM\_SHA512\_256\_HMAC\_GENERAL**

```
#define CKM_SHA512_256_HMAC_GENERAL 0x00000004EUL
```

**20.171.1.530 CKM\_SHA512\_256\_KEY\_DERIVATION**

```
#define CKM_SHA512_256_KEY_DERIVATION 0x00000004FUL
```

### 20.171.1.531 CKM\_SHA512\_HMAC

```
#define CKM_SHA512_HMAC 0x00000271UL
```

### 20.171.1.532 CKM\_SHA512\_HMAC\_GENERAL

```
#define CKM_SHA512_HMAC_GENERAL 0x00000272UL
```

### 20.171.1.533 CKM\_SHA512\_KEY\_DERIVATION

```
#define CKM_SHA512_KEY_DERIVATION 0x00000395UL
```

### 20.171.1.534 CKM\_SHA512\_RSA\_PKCS

```
#define CKM_SHA512_RSA_PKCS 0x00000042UL
```

### 20.171.1.535 CKM\_SHA512\_RSA\_PKCS\_PSS

```
#define CKM_SHA512_RSA_PKCS_PSS 0x00000045UL
```

### 20.171.1.536 CKM\_SHA512\_T

```
#define CKM_SHA512_T 0x00000050UL
```

### 20.171.1.537 CKM\_SHA512\_T\_HMAC

```
#define CKM_SHA512_T_HMAC 0x00000051UL
```

### 20.171.1.538 CKM\_SHA512\_T\_HMAC\_GENERAL

```
#define CKM_SHA512_T_HMAC_GENERAL 0x00000052UL
```

**20.171.1.539 CKM\_SHA512\_T\_KEY\_DERIVATION**

```
#define CKM_SHA512_T_KEY_DERIVATION 0x00000053UL
```

**20.171.1.540 CKM\_SHA\_1**

```
#define CKM_SHA_1 0x000000220UL
```

**20.171.1.541 CKM\_SHA\_1\_HMAC**

```
#define CKM_SHA_1_HMAC 0x000000221UL
```

**20.171.1.542 CKM\_SHA\_1\_HMAC\_GENERAL**

```
#define CKM_SHA_1_HMAC_GENERAL 0x000000222UL
```

**20.171.1.543 CKM\_SKIPJACK\_CBC64**

```
#define CKM_SKIPJACK_CBC64 0x00001002UL
```

**20.171.1.544 CKM\_SKIPJACK\_CFB16**

```
#define CKM_SKIPJACK_CFB16 0x00001006UL
```

**20.171.1.545 CKM\_SKIPJACK\_CFB32**

```
#define CKM_SKIPJACK_CFB32 0x00001005UL
```

**20.171.1.546 CKM\_SKIPJACK\_CFB64**

```
#define CKM_SKIPJACK_CFB64 0x00001004UL
```

### 20.171.1.547 CKM\_SKIPJACK\_CFB8

```
#define CKM_SKIPJACK_CFB8 0x00001007UL
```

### 20.171.1.548 CKM\_SKIPJACK\_ECB64

```
#define CKM_SKIPJACK_ECB64 0x00001001UL
```

### 20.171.1.549 CKM\_SKIPJACK\_KEY\_GEN

```
#define CKM_SKIPJACK_KEY_GEN 0x00001000UL
```

### 20.171.1.550 CKM\_SKIPJACK\_OFB64

```
#define CKM_SKIPJACK_OFB64 0x00001003UL
```

### 20.171.1.551 CKM\_SKIPJACK\_PRIVATE\_WRAP

```
#define CKM_SKIPJACK_PRIVATE_WRAP 0x00001009UL
```

### 20.171.1.552 CKM\_SKIPJACK\_RELAYX

```
#define CKM_SKIPJACK_RELAYX 0x0000100aUL
```

### 20.171.1.553 CKM\_SKIPJACK\_WRAP

```
#define CKM_SKIPJACK_WRAP 0x00001008UL
```

### 20.171.1.554 CKM\_SSL3\_KEY\_AND\_MAC\_DERIVE

```
#define CKM_SSL3_KEY_AND_MAC_DERIVE 0x00000372UL
```

**20.171.1.555 CKM\_SSL3\_MASTER\_KEY\_DERIVE**

```
#define CKM_SSL3_MASTER_KEY_DERIVE 0x00000371UL
```

**20.171.1.556 CKM\_SSL3\_MASTER\_KEY\_DERIVE\_DH**

```
#define CKM_SSL3_MASTER_KEY_DERIVE_DH 0x00000373UL
```

**20.171.1.557 CKM\_SSL3\_MD5\_MAC**

```
#define CKM_SSL3_MD5_MAC 0x00000380UL
```

**20.171.1.558 CKM\_SSL3\_PRE\_MASTER\_KEY\_GEN**

```
#define CKM_SSL3_PRE_MASTER_KEY_GEN 0x00000370UL
```

**20.171.1.559 CKM\_SSL3\_SHA1\_MAC**

```
#define CKM_SSL3_SHA1_MAC 0x00000381UL
```

**20.171.1.560 CKM\_TLS10\_MAC\_CLIENT**

```
#define CKM_TLS10_MAC_CLIENT 0x000003D7UL
```

**20.171.1.561 CKM\_TLS10\_MAC\_SERVER**

```
#define CKM_TLS10_MAC_SERVER 0x000003D6UL
```

**20.171.1.562 CKM\_TLS12\_KDF**

```
#define CKM_TLS12_KDF 0x000003D9UL
```

### 20.171.1.563 CKM\_TLS12\_KEY\_AND\_MAC\_DERIVE

```
#define CKM_TLS12_KEY_AND_MAC_DERIVE 0x000003E1UL
```

### 20.171.1.564 CKM\_TLS12\_KEY\_SAFE\_DERIVE

```
#define CKM_TLS12_KEY_SAFE_DERIVE 0x000003E3UL
```

### 20.171.1.565 CKM\_TLS12\_MAC

```
#define CKM_TLS12_MAC 0x000003D8UL
```

### 20.171.1.566 CKM\_TLS12\_MASTER\_KEY\_DERIVE

```
#define CKM_TLS12_MASTER_KEY_DERIVE 0x000003E0UL
```

### 20.171.1.567 CKM\_TLS12\_MASTER\_KEY\_DERIVE\_DH

```
#define CKM_TLS12_MASTER_KEY_DERIVE_DH 0x000003E2UL
```

### 20.171.1.568 CKM\_TLS\_KDF

```
#define CKM_TLS_KDF 0x000003E5UL
```

### 20.171.1.569 CKM\_TLS\_KEY\_AND\_MAC\_DERIVE

```
#define CKM_TLS_KEY_AND_MAC_DERIVE 0x00000376UL
```

### 20.171.1.570 CKM\_TLS\_MAC

```
#define CKM_TLS_MAC 0x000003E4UL
```

**20.171.1.571 CKM\_TLS\_MASTER\_KEY\_DERIVE**

```
#define CKM_TLS_MASTER_KEY_DERIVE 0x00000375UL
```

**20.171.1.572 CKM\_TLS\_MASTER\_KEY\_DERIVE\_DH**

```
#define CKM_TLS_MASTER_KEY_DERIVE_DH 0x00000377UL
```

**20.171.1.573 CKM\_TLS\_PRE\_MASTER\_KEY\_GEN**

```
#define CKM_TLS_PRE_MASTER_KEY_GEN 0x00000374UL
```

**20.171.1.574 CKM\_TLS\_PRF**

```
#define CKM_TLS_PRF 0x00000378UL
```

**20.171.1.575 CKM\_TWOFISH\_CBC**

```
#define CKM_TWOFISH_CBC 0x00001093UL
```

**20.171.1.576 CKM\_TWOFISH\_CBC\_PAD**

```
#define CKM_TWOFISH_CBC_PAD 0x00001095UL
```

**20.171.1.577 CKM\_TWOFISH\_KEY\_GEN**

```
#define CKM_TWOFISH_KEY_GEN 0x00001092UL
```

**20.171.1.578 CKM\_VENDOR\_DEFINED**

```
#define CKM_VENDOR_DEFINED 0x80000000UL
```

### 20.171.1.579 CKM\_WTLS\_CLIENT\_KEY\_AND\_MAC\_DERIVE

```
#define CKM_WTLS_CLIENT_KEY_AND_MAC_DERIVE 0x000003D5UL
```

### 20.171.1.580 CKM\_WTLS\_MASTER\_KEY\_DERIVE

```
#define CKM_WTLS_MASTER_KEY_DERIVE 0x000003D1UL
```

### 20.171.1.581 CKM\_WTLS\_MASTER\_KEY\_DERIVE\_DH\_ECC

```
#define CKM_WTLS_MASTER_KEY_DERIVE_DH_ECC 0x000003D2UL
```

### 20.171.1.582 CKM\_WTLS\_PRE\_MASTER\_KEY\_GEN

```
#define CKM_WTLS_PRE_MASTER_KEY_GEN 0x000003D0UL
```

### 20.171.1.583 CKM\_WTLS\_PRF

```
#define CKM_WTLS_PRF 0x000003D3UL
```

### 20.171.1.584 CKM\_WTLS\_SERVER\_KEY\_AND\_MAC\_DERIVE

```
#define CKM_WTLS_SERVER_KEY_AND_MAC_DERIVE 0x000003D4UL
```

### 20.171.1.585 CKM\_X9\_42\_DH\_DERIVE

```
#define CKM_X9_42_DH_DERIVE 0x00000031UL
```

### 20.171.1.586 CKM\_X9\_42\_DH\_HYBRID\_DERIVE

```
#define CKM_X9_42_DH_HYBRID_DERIVE 0x00000032UL
```



**20.171.1.587 CKM\_X9\_42\_DH\_KEY\_PAIR\_GEN**

```
#define CKM_X9_42_DH_KEY_PAIR_GEN 0x00000030UL
```

**20.171.1.588 CKM\_X9\_42\_DH\_PARAMETER\_GEN**

```
#define CKM_X9_42_DH_PARAMETER_GEN 0x00002002UL
```

**20.171.1.589 CKM\_X9\_42\_MQV\_DERIVE**

```
#define CKM_X9_42_MQV_DERIVE 0x00000033UL
```

**20.171.1.590 CKM\_XOR\_BASE\_AND\_DATA**

```
#define CKM_XOR_BASE_AND_DATA 0x00000364UL
```

**20.171.1.591 CKN\_OTP\_CHANGED**

```
#define CKN_OTP_CHANGED 1UL
```

**20.171.1.592 CKN\_SURRENDER**

```
#define CKN_SURRENDER 0UL
```

**20.171.1.593 CKO\_CERTIFICATE**

```
#define CKO_CERTIFICATE 0x00000001UL
```

**20.171.1.594 CKO\_DATA**

```
#define CKO_DATA 0x00000000UL
```

### 20.171.1.595 CKO\_DOMAIN\_PARAMETERS

```
#define CKO_DOMAIN_PARAMETERS 0x00000006UL
```

### 20.171.1.596 CKO\_HW\_FEATURE

```
#define CKO_HW_FEATURE 0x00000005UL
```

### 20.171.1.597 CKO\_MECHANISM

```
#define CKO_MECHANISM 0x00000007UL
```

### 20.171.1.598 CKO\_OTP\_KEY

```
#define CKO_OTP_KEY 0x00000008UL
```

### 20.171.1.599 CKO\_PRIVATE\_KEY

```
#define CKO_PRIVATE_KEY 0x00000003UL
```

### 20.171.1.600 CKO\_PUBLIC\_KEY

```
#define CKO_PUBLIC_KEY 0x00000002UL
```

### 20.171.1.601 CKO\_SECRET\_KEY

```
#define CKO_SECRET_KEY 0x00000004UL
```

### 20.171.1.602 CKO\_VENDOR\_DEFINED

```
#define CKO_VENDOR_DEFINED 0x80000000UL
```

**20.171.1.603 CKP\_PKCS5\_PBKD2\_HMAC\_GOSTR3411**

```
#define CKP_PKCS5_PBKD2_HMAC_GOSTR3411 0x00000002UL
```

**20.171.1.604 CKP\_PKCS5\_PBKD2\_HMAC\_SHA1**

```
#define CKP_PKCS5_PBKD2_HMAC_SHA1 0x00000001UL
```

**20.171.1.605 CKP\_PKCS5\_PBKD2\_HMAC\_SHA224**

```
#define CKP_PKCS5_PBKD2_HMAC_SHA224 0x00000003UL
```

**20.171.1.606 CKP\_PKCS5\_PBKD2\_HMAC\_SHA256**

```
#define CKP_PKCS5_PBKD2_HMAC_SHA256 0x00000004UL
```

**20.171.1.607 CKP\_PKCS5\_PBKD2\_HMAC\_SHA384**

```
#define CKP_PKCS5_PBKD2_HMAC_SHA384 0x00000005UL
```

**20.171.1.608 CKP\_PKCS5\_PBKD2\_HMAC\_SHA512**

```
#define CKP_PKCS5_PBKD2_HMAC_SHA512 0x00000006UL
```

**20.171.1.609 CKP\_PKCS5\_PBKD2\_HMAC\_SHA512\_224**

```
#define CKP_PKCS5_PBKD2_HMAC_SHA512_224 0x00000007UL
```

**20.171.1.610 CKP\_PKCS5\_PBKD2\_HMAC\_SHA512\_256**

```
#define CKP_PKCS5_PBKD2_HMAC_SHA512_256 0x00000008UL
```

### 20.171.1.611 CKR\_ACTION\_PROHIBITED

```
#define CKR_ACTION_PROHIBITED 0x0000001BUL
```

### 20.171.1.612 CKR\_ARGUMENTS\_BAD

```
#define CKR_ARGUMENTS_BAD 0x00000007UL
```

### 20.171.1.613 CKR\_ATTRIBUTE\_READ\_ONLY

```
#define CKR_ATTRIBUTE_READ_ONLY 0x00000010UL
```

### 20.171.1.614 CKR\_ATTRIBUTE\_SENSITIVE

```
#define CKR_ATTRIBUTE_SENSITIVE 0x00000011UL
```

### 20.171.1.615 CKR\_ATTRIBUTE\_TYPE\_INVALID

```
#define CKR_ATTRIBUTE_TYPE_INVALID 0x00000012UL
```

### 20.171.1.616 CKR\_ATTRIBUTE\_VALUE\_INVALID

```
#define CKR_ATTRIBUTE_VALUE_INVALID 0x00000013UL
```

### 20.171.1.617 CKR\_BUFFER\_TOO\_SMALL

```
#define CKR_BUFFER_TOO_SMALL 0x00000150UL
```

### 20.171.1.618 CKR\_CANCEL

```
#define CKR_CANCEL 0x00000001UL
```

**20.171.1.619 CKR\_CANT\_LOCK**

```
#define CKR_CANT_LOCK 0x0000000AUL
```

**20.171.1.620 CKR\_CRYPTOKI\_ALREADY\_INITIALIZED**

```
#define CKR_CRYPTOKI_ALREADY_INITIALIZED 0x00000191UL
```

**20.171.1.621 CKR\_CRYPTOKI\_NOT\_INITIALIZED**

```
#define CKR_CRYPTOKI_NOT_INITIALIZED 0x00000190UL
```

**20.171.1.622 CKR\_CURVE\_NOT\_SUPPORTED**

```
#define CKR_CURVE_NOT_SUPPORTED 0x00000140UL
```

**20.171.1.623 CKR\_DATA\_INVALID**

```
#define CKR_DATA_INVALID 0x00000020UL
```

**20.171.1.624 CKR\_DATA\_LEN\_RANGE**

```
#define CKR_DATA_LEN_RANGE 0x00000021UL
```

**20.171.1.625 CKR\_DEVICE\_ERROR**

```
#define CKR_DEVICE_ERROR 0x00000030UL
```

**20.171.1.626 CKR\_DEVICE\_MEMORY**

```
#define CKR_DEVICE_MEMORY 0x00000031UL
```

### 20.171.1.627 CKR\_DEVICE\_REMOVED

```
#define CKR_DEVICE_REMOVED 0x00000032UL
```

### 20.171.1.628 CKR\_DOMAIN\_PARAMS\_INVALID

```
#define CKR_DOMAIN_PARAMS_INVALID 0x00000130UL
```

### 20.171.1.629 CKR\_ENCRYPTED\_DATA\_INVALID

```
#define CKR_ENCRYPTED_DATA_INVALID 0x00000040UL
```

### 20.171.1.630 CKR\_ENCRYPTED\_DATA\_LEN\_RANGE

```
#define CKR_ENCRYPTED_DATA_LEN_RANGE 0x00000041UL
```

### 20.171.1.631 CKR\_EXCEEDED\_MAX\_ITERATIONS

```
#define CKR_EXCEEDED_MAX_ITERATIONS 0x000001B5UL
```

### 20.171.1.632 CKR\_FIPS\_SELF\_TEST\_FAILED

```
#define CKR_FIPS_SELF_TEST_FAILED 0x000001B6UL
```

### 20.171.1.633 CKR\_FUNCTION\_CANCELED

```
#define CKR_FUNCTION_CANCELED 0x00000050UL
```

### 20.171.1.634 CKR\_FUNCTION\_FAILED

```
#define CKR_FUNCTION_FAILED 0x00000006UL
```

**20.171.1.635 CKR\_FUNCTION\_NOT\_PARALLEL**

```
#define CKR_FUNCTION_NOT_PARALLEL 0x00000051UL
```

**20.171.1.636 CKR\_FUNCTION\_NOT\_SUPPORTED**

```
#define CKR_FUNCTION_NOT_SUPPORTED 0x00000054UL
```

**20.171.1.637 CKR\_FUNCTION\_REJECTED**

```
#define CKR_FUNCTION_REJECTED 0x00000200UL
```

**20.171.1.638 CKR\_GENERAL\_ERROR**

```
#define CKR_GENERAL_ERROR 0x00000005UL
```

**20.171.1.639 CKR\_HOST\_MEMORY**

```
#define CKR_HOST_MEMORY 0x00000002UL
```

**20.171.1.640 CKR\_INFORMATION\_SENSITIVE**

```
#define CKR_INFORMATION_SENSITIVE 0x00000170UL
```

**20.171.1.641 CKR\_KEY\_CHANGED**

```
#define CKR_KEY_CHANGED 0x00000065UL
```

**20.171.1.642 CKR\_KEY\_FUNCTION\_NOT\_PERMITTED**

```
#define CKR_KEY_FUNCTION_NOT_PERMITTED 0x00000068UL
```

### 20.171.1.643 CKR\_KEY\_HANDLE\_INVALID

```
#define CKR_KEY_HANDLE_INVALID 0x00000060UL
```

### 20.171.1.644 CKR\_KEY\_INDIGESTIBLE

```
#define CKR_KEY_INDIGESTIBLE 0x00000067UL
```

### 20.171.1.645 CKR\_KEY\_NEEDED

```
#define CKR_KEY_NEEDED 0x00000066UL
```

### 20.171.1.646 CKR\_KEY\_NOT\_NEEDED

```
#define CKR_KEY_NOT_NEEDED 0x00000064UL
```

### 20.171.1.647 CKR\_KEY\_NOT\_WRAPABLE

```
#define CKR_KEY_NOT_WRAPABLE 0x00000069UL
```

### 20.171.1.648 CKR\_KEY\_SIZE\_RANGE

```
#define CKR_KEY_SIZE_RANGE 0x00000062UL
```

### 20.171.1.649 CKR\_KEY\_TYPE\_INCONSISTENT

```
#define CKR_KEY_TYPE_INCONSISTENT 0x00000063UL
```

### 20.171.1.650 CKR\_KEY\_UNEXTRACTABLE

```
#define CKR_KEY_UNEXTRACTABLE 0x0000006AUL
```



**20.171.1.651 CKR\_LIBRARY\_LOAD\_FAILED**

```
#define CKR_LIBRARY_LOAD_FAILED 0x000001B7UL
```

**20.171.1.652 CKR\_MECHANISM\_INVALID**

```
#define CKR_MECHANISM_INVALID 0x00000070UL
```

**20.171.1.653 CKR\_MECHANISM\_PARAM\_INVALID**

```
#define CKR_MECHANISM_PARAM_INVALID 0x00000071UL
```

**20.171.1.654 CKR\_MUTEX\_BAD**

```
#define CKR_MUTEX_BAD 0x000001A0UL
```

**20.171.1.655 CKR\_MUTEX\_NOT\_LOCKED**

```
#define CKR_MUTEX_NOT_LOCKED 0x000001A1UL
```

**20.171.1.656 CKR\_NEED\_TO\_CREATE\_THREADS**

```
#define CKR_NEED_TO_CREATE_THREADS 0x00000009UL
```

**20.171.1.657 CKR\_NEW\_PIN\_MODE**

```
#define CKR_NEW_PIN_MODE 0x000001B0UL
```

**20.171.1.658 CKR\_NEXT\_OTP**

```
#define CKR_NEXT_OTP 0x000001B1UL
```

### 20.171.1.659 CKR\_NO\_EVENT

```
#define CKR_NO_EVENT 0x00000008UL
```

### 20.171.1.660 CKR\_OBJECT\_HANDLE\_INVALID

```
#define CKR_OBJECT_HANDLE_INVALID 0x00000082UL
```

### 20.171.1.661 CKR\_OK

```
#define CKR_OK 0x00000000UL
```

### 20.171.1.662 CKR\_OPERATION\_ACTIVE

```
#define CKR_OPERATION_ACTIVE 0x00000090UL
```

### 20.171.1.663 CKR\_OPERATION\_NOT\_INITIALIZED

```
#define CKR_OPERATION_NOT_INITIALIZED 0x00000091UL
```

### 20.171.1.664 CKR\_PIN\_EXPIRED

```
#define CKR_PIN_EXPIRED 0x000000A3UL
```

### 20.171.1.665 CKR\_PIN\_INCORRECT

```
#define CKR_PIN_INCORRECT 0x000000A0UL
```

### 20.171.1.666 CKR\_PIN\_INVALID

```
#define CKR_PIN_INVALID 0x000000A1UL
```

**20.171.1.667 CKR\_PIN\_LEN\_RANGE**

```
#define CKR_PIN_LEN_RANGE 0x000000A2UL
```

**20.171.1.668 CKR\_PIN\_LOCKED**

```
#define CKR_PIN_LOCKED 0x000000A4UL
```

**20.171.1.669 CKR\_PIN\_TOO\_WEAK**

```
#define CKR_PIN_TOO_WEAK 0x000001B8UL
```

**20.171.1.670 CKR\_PUBLIC\_KEY\_INVALID**

```
#define CKR_PUBLIC_KEY_INVALID 0x000001B9UL
```

**20.171.1.671 CKR\_RANDOM\_NO\_RNG**

```
#define CKR_RANDOM_NO_RNG 0x00000121UL
```

**20.171.1.672 CKR\_RANDOM\_SEED\_NOT\_SUPPORTED**

```
#define CKR_RANDOM_SEED_NOT_SUPPORTED 0x00000120UL
```

**20.171.1.673 CKR\_SAVED\_STATE\_INVALID**

```
#define CKR_SAVED_STATE_INVALID 0x00000160UL
```

**20.171.1.674 CKR\_SESSION\_CLOSED**

```
#define CKR_SESSION_CLOSED 0x000000B0UL
```

### 20.171.1.675 CKR\_SESSION\_COUNT

```
#define CKR_SESSION_COUNT 0x000000B1UL
```

### 20.171.1.676 CKR\_SESSION\_EXISTS

```
#define CKR_SESSION_EXISTS 0x000000B6UL
```

### 20.171.1.677 CKR\_SESSION\_HANDLE\_INVALID

```
#define CKR_SESSION_HANDLE_INVALID 0x000000B3UL
```

### 20.171.1.678 CKR\_SESSION\_PARALLEL\_NOT\_SUPPORTED

```
#define CKR_SESSION_PARALLEL_NOT_SUPPORTED 0x000000B4UL
```

### 20.171.1.679 CKR\_SESSION\_READ\_ONLY

```
#define CKR_SESSION_READ_ONLY 0x000000B5UL
```

### 20.171.1.680 CKR\_SESSION\_READ\_ONLY\_EXISTS

```
#define CKR_SESSION_READ_ONLY_EXISTS 0x000000B7UL
```

### 20.171.1.681 CKR\_SESSION\_READ\_WRITE\_SO\_EXISTS

```
#define CKR_SESSION_READ_WRITE_SO_EXISTS 0x000000B8UL
```

### 20.171.1.682 CKR\_SIGNATURE\_INVALID

```
#define CKR_SIGNATURE_INVALID 0x000000C0UL
```

**20.171.1.683 CKR\_SIGNATURE\_LEN\_RANGE**

```
#define CKR_SIGNATURE_LEN_RANGE 0x000000C1UL
```

**20.171.1.684 CKR\_SLOT\_ID\_INVALID**

```
#define CKR_SLOT_ID_INVALID 0x00000003UL
```

**20.171.1.685 CKR\_STATE\_UNSAVEABLE**

```
#define CKR_STATE_UNSAVEABLE 0x00000180UL
```

**20.171.1.686 CKR\_TEMPLATE\_INCOMPLETE**

```
#define CKR_TEMPLATE_INCOMPLETE 0x000000D0UL
```

**20.171.1.687 CKR\_TEMPLATE\_INCONSISTENT**

```
#define CKR_TEMPLATE_INCONSISTENT 0x000000D1UL
```

**20.171.1.688 CKR\_TOKEN\_NOT\_PRESENT**

```
#define CKR_TOKEN_NOT_PRESENT 0x000000E0UL
```

**20.171.1.689 CKR\_TOKEN\_NOT\_RECOGNIZED**

```
#define CKR_TOKEN_NOT_RECOGNIZED 0x000000E1UL
```

**20.171.1.690 CKR\_TOKEN\_WRITE\_PROTECTED**

```
#define CKR_TOKEN_WRITE_PROTECTED 0x000000E2UL
```

### 20.171.1.691 CKR\_UNWRAPPING\_KEY\_HANDLE\_INVALID

```
#define CKR_UNWRAPPING_KEY_HANDLE_INVALID 0x000000F0UL
```

### 20.171.1.692 CKR\_UNWRAPPING\_KEY\_SIZE\_RANGE

```
#define CKR_UNWRAPPING_KEY_SIZE_RANGE 0x000000F1UL
```

### 20.171.1.693 CKR\_UNWRAPPING\_KEY\_TYPE\_INCONSISTENT

```
#define CKR_UNWRAPPING_KEY_TYPE_INCONSISTENT 0x000000F2UL
```

### 20.171.1.694 CKR\_USER\_ALREADY\_LOGGED\_IN

```
#define CKR_USER_ALREADY_LOGGED_IN 0x00000100UL
```

### 20.171.1.695 CKR\_USER\_ANOTHER\_ALREADY\_LOGGED\_IN

```
#define CKR_USER_ANOTHER_ALREADY_LOGGED_IN 0x00000104UL
```

### 20.171.1.696 CKR\_USER\_NOT\_LOGGED\_IN

```
#define CKR_USER_NOT_LOGGED_IN 0x00000101UL
```

### 20.171.1.697 CKR\_USER\_PIN\_NOT\_INITIALIZED

```
#define CKR_USER_PIN_NOT_INITIALIZED 0x00000102UL
```

### 20.171.1.698 CKR\_USER\_TOO\_MANY\_TYPES

```
#define CKR_USER_TOO_MANY_TYPES 0x00000105UL
```

**20.171.1.699 CKR\_USER\_TYPE\_INVALID**

```
#define CKR_USER_TYPE_INVALID 0x00000103UL
```

**20.171.1.700 CKR\_VENDOR\_DEFINED**

```
#define CKR_VENDOR_DEFINED 0x80000000UL
```

**20.171.1.701 CKR\_WRAPPED\_KEY\_INVALID**

```
#define CKR_WRAPPED_KEY_INVALID 0x00000110UL
```

**20.171.1.702 CKR\_WRAPPED\_KEY\_LEN\_RANGE**

```
#define CKR_WRAPPED_KEY_LEN_RANGE 0x00000112UL
```

**20.171.1.703 CKR\_WRAPPING\_KEY\_HANDLE\_INVALID**

```
#define CKR_WRAPPING_KEY_HANDLE_INVALID 0x00000113UL
```

**20.171.1.704 CKR\_WRAPPING\_KEY\_SIZE\_RANGE**

```
#define CKR_WRAPPING_KEY_SIZE_RANGE 0x00000114UL
```

**20.171.1.705 CKR\_WRAPPING\_KEY\_TYPE\_INCONSISTENT**

```
#define CKR_WRAPPING_KEY_TYPE_INCONSISTENT 0x00000115UL
```

**20.171.1.706 CKS\_RO\_PUBLIC\_SESSION**

```
#define CKS_RO_PUBLIC_SESSION 0UL
```

### 20.171.1.707 CKS\_RO\_USER\_FUNCTIONS

```
#define CKS_RO_USER_FUNCTIONS 1UL
```

### 20.171.1.708 CKS\_RW\_PUBLIC\_SESSION

```
#define CKS_RW_PUBLIC_SESSION 2UL
```

### 20.171.1.709 CKS\_RW\_SO\_FUNCTIONS

```
#define CKS_RW_SO_FUNCTIONS 4UL
```

### 20.171.1.710 CKS\_RW\_USER\_FUNCTIONS

```
#define CKS_RW_USER_FUNCTIONS 3UL
```

### 20.171.1.711 CKU\_CONTEXT\_SPECIFIC

```
#define CKU_CONTEXT_SPECIFIC 2UL
```

### 20.171.1.712 CKU\_SO

```
#define CKU_SO 0UL
```

### 20.171.1.713 CKU\_USER

```
#define CKU_USER 1UL
```

### 20.171.1.714 CKZ\_DATA\_SPECIFIED

```
#define CKZ_DATA_SPECIFIED 0x00000001UL
```



**20.171.1.715 CKZ\_SALT\_SPECIFIED**

```
#define CKZ_SALT_SPECIFIED 0x00000001UL
```

**20.171.1.716 CRYPTOKI\_VERSION\_AMENDMENT**

```
#define CRYPTOKI_VERSION_AMENDMENT 0
```

**20.171.1.717 CRYPTOKI\_VERSION\_MAJOR**

```
#define CRYPTOKI_VERSION_MAJOR 2
```

**20.171.1.718 CRYPTOKI\_VERSION\_MINOR**

```
#define CRYPTOKI_VERSION_MINOR 40
```

**20.171.1.719 FALSE**

```
#define FALSE CK\_FALSE
```

**20.171.1.720 TRUE**

```
#define TRUE CK\_TRUE
```

**20.171.2 Typedef Documentation****20.171.2.1 CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS**

```
typedef struct CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS
```

### 20.171.2.2 CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR

```
typedef CK_AES_CBC_ENCRYPT_DATA_PARAMS CK_PTR CK_AES_CBC_ENCRYPT_DATA_PARAMS_PTR
```

### 20.171.2.3 CK\_AES\_CCM\_PARAMS

```
typedef struct CK_AES_CCM_PARAMS CK_AES_CCM_PARAMS
```

### 20.171.2.4 CK\_AES\_CCM\_PARAMS\_PTR

```
typedef CK_AES_CCM_PARAMS CK_PTR CK_AES_CCM_PARAMS_PTR
```

### 20.171.2.5 CK\_AES\_CTR\_PARAMS

```
typedef struct CK_AES_CTR_PARAMS CK_AES_CTR_PARAMS
```

### 20.171.2.6 CK\_AES\_CTR\_PARAMS\_PTR

```
typedef CK_AES_CTR_PARAMS CK_PTR CK_AES_CTR_PARAMS_PTR
```

### 20.171.2.7 CK\_AES\_GCM\_PARAMS

```
typedef struct CK_AES_GCM_PARAMS CK_AES_GCM_PARAMS
```

### 20.171.2.8 CK\_AES\_GCM\_PARAMS\_PTR

```
typedef CK_AES_GCM_PARAMS CK_PTR CK_AES_GCM_PARAMS_PTR
```

### 20.171.2.9 CK\_ARIA\_CBC\_ENCRYPT\_DATA\_PARAMS

```
typedef struct CK_ARIA_CBC_ENCRYPT_DATA_PARAMS CK_ARIA_CBC_ENCRYPT_DATA_PARAMS
```

**20.171.2.10 CK\_ARIA\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR**

```
typedef CK_ARIA_CBC_ENCRYPT_DATA_PARAMS CK_PTR CK_ARIA_CBC_ENCRYPT_DATA_PARAMS_PTR
```

**20.171.2.11 CK\_ATTRIBUTE**

```
typedef struct CK_ATTRIBUTE CK_ATTRIBUTE
```

**20.171.2.12 CK\_ATTRIBUTE\_PTR**

```
typedef CK_ATTRIBUTE CK_PTR CK_ATTRIBUTE_PTR
```

**20.171.2.13 CK\_ATTRIBUTE\_TYPE**

```
typedef CK_ULONG CK_ATTRIBUTE_TYPE
```

**20.171.2.14 CK\_BBOOL**

```
typedef CK_BYTE CK_BBOOL
```

**20.171.2.15 CK\_BYTE**

```
typedef unsigned char CK_BYTE
```

**20.171.2.16 CK\_BYTE\_PTR**

```
typedef CK_BYTE CK_PTR CK_BYTE_PTR
```

**20.171.2.17 CK\_C\_INITIALIZE\_ARGS**

```
typedef struct CK_C_INITIALIZE_ARGS CK_C_INITIALIZE_ARGS
```

### 20.171.2.18 CK\_C\_INITIALIZE\_ARGS\_PTR

```
typedef CK_C_INITIALIZE_ARGS CK_PTR CK_C_INITIALIZE_ARGS_PTR
```

### 20.171.2.19 CK\_CAMELLIA\_CBC\_ENCRYPT\_DATA\_PARAMS

```
typedef struct CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS
```

### 20.171.2.20 CK\_CAMELLIA\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR

```
typedef CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS CK_PTR CK_CAMELLIA_CBC_ENCRYPT_DATA_PARAMS_PTR
```

### 20.171.2.21 CK\_CAMELLIA\_CTR\_PARAMS

```
typedef struct CK_CAMELLIA_CTR_PARAMS CK_CAMELLIA_CTR_PARAMS
```

### 20.171.2.22 CK\_CAMELLIA\_CTR\_PARAMS\_PTR

```
typedef CK_CAMELLIA_CTR_PARAMS CK_PTR CK_CAMELLIA_CTR_PARAMS_PTR
```

### 20.171.2.23 CK\_CCM\_PARAMS

```
typedef struct CK_CCM_PARAMS CK_CCM_PARAMS
```

### 20.171.2.24 CK\_CCM\_PARAMS\_PTR

```
typedef CK_CCM_PARAMS CK_PTR CK_CCM_PARAMS_PTR
```

### 20.171.2.25 CK\_CERTIFICATE\_CATEGORY

```
typedef CK_ULONG CK_CERTIFICATE_CATEGORY
```

**20.171.2.26 CK\_CERTIFICATE\_TYPE**

```
typedef CK_ULONG CK_CERTIFICATE_TYPE
```

**20.171.2.27 CK\_CHAR**

```
typedef CK_BYTE CK_CHAR
```

**20.171.2.28 CK\_CHAR\_PTR**

```
typedef CK_CHAR CK_PTR CK_CHAR_PTR
```

**20.171.2.29 CK\_CMS\_SIG\_PARAMS**

```
typedef struct CK_CMS_SIG_PARAMS CK_CMS_SIG_PARAMS
```

**20.171.2.30 CK\_CMS\_SIG\_PARAMS\_PTR**

```
typedef CK_CMS_SIG_PARAMS CK_PTR CK_CMS_SIG_PARAMS_PTR
```

**20.171.2.31 CK\_DATE**

```
typedef struct CK_DATE CK_DATE
```

**20.171.2.32 CK\_DES\_CBC\_ENCRYPT\_DATA\_PARAMS**

```
typedef struct CK_DES_CBC_ENCRYPT_DATA_PARAMS CK_DES_CBC_ENCRYPT_DATA_PARAMS
```

**20.171.2.33 CK\_DES\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR**

```
typedef CK_DES_CBC_ENCRYPT_DATA_PARAMS CK_PTR CK_DES_CBC_ENCRYPT_DATA_PARAMS_PTR
```

### 20.171.2.34 CK\_DSA\_PARAMETER\_GEN\_PARAM

```
typedef struct CK_DSA_PARAMETER_GEN_PARAM CK_DSA_PARAMETER_GEN_PARAM
```

### 20.171.2.35 CK\_DSA\_PARAMETER\_GEN\_PARAM\_PTR

```
typedef CK_DSA_PARAMETER_GEN_PARAM CK_PTR CK_DSA_PARAMETER_GEN_PARAM_PTR
```

### 20.171.2.36 CK\_EC\_KDF\_TYPE

```
typedef CK_ULONG CK_EC_KDF_TYPE
```

### 20.171.2.37 CK\_ECDH1\_DERIVE\_PARAMS

```
typedef struct CK_ECDH1_DERIVE_PARAMS CK_ECDH1_DERIVE_PARAMS
```

### 20.171.2.38 CK\_ECDH1\_DERIVE\_PARAMS\_PTR

```
typedef CK_ECDH1_DERIVE_PARAMS CK_PTR CK_ECDH1_DERIVE_PARAMS_PTR
```

### 20.171.2.39 CK\_ECDH2\_DERIVE\_PARAMS

```
typedef struct CK_ECDH2_DERIVE_PARAMS CK_ECDH2_DERIVE_PARAMS
```

### 20.171.2.40 CK\_ECDH2\_DERIVE\_PARAMS\_PTR

```
typedef CK_ECDH2_DERIVE_PARAMS CK_PTR CK_ECDH2_DERIVE_PARAMS_PTR
```

### 20.171.2.41 CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS

```
typedef struct CK_ECDH_AES_KEY_WRAP_PARAMS CK_ECDH_AES_KEY_WRAP_PARAMS
```

**20.171.2.42 CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS\_PTR**

```
typedef CK_ECDH_AES_KEY_WRAP_PARAMS CK_PTR CK_ECDH_AES_KEY_WRAP_PARAMS_PTR
```

**20.171.2.43 CK\_ECMQV\_DERIVE\_PARAMS**

```
typedef struct CK_ECMQV_DERIVE_PARAMS CK_ECMQV_DERIVE_PARAMS
```

**20.171.2.44 CK\_ECMQV\_DERIVE\_PARAMS\_PTR**

```
typedef CK_ECMQV_DERIVE_PARAMS CK_PTR CK_ECMQV_DERIVE_PARAMS_PTR
```

**20.171.2.45 CK\_EXTRACT\_PARAMS**

```
typedef CK_ULONG CK_EXTRACT_PARAMS
```

**20.171.2.46 CK\_EXTRACT\_PARAMS\_PTR**

```
typedef CK_EXTRACT_PARAMS CK_PTR CK_EXTRACT_PARAMS_PTR
```

**20.171.2.47 CK\_FLAGS**

```
typedef CK_ULONG CK_FLAGS
```

**20.171.2.48 CK\_FUNCTION\_LIST**

```
typedef struct CK_FUNCTION_LIST CK_FUNCTION_LIST
```

**20.171.2.49 CK\_FUNCTION\_LIST\_PTR**

```
typedef CK_FUNCTION_LIST CK_PTR CK_FUNCTION_LIST_PTR
```

### 20.171.2.50 CK\_FUNCTION\_LIST\_PTR\_PTR

```
typedef CK_FUNCTION_LIST_PTR CK_PTR CK_FUNCTION_LIST_PTR_PTR
```

### 20.171.2.51 CK\_GCM\_PARAMS

```
typedef struct CK_GCM_PARAMS CK_GCM_PARAMS
```

### 20.171.2.52 CK\_GCM\_PARAMS\_PTR

```
typedef CK_GCM_PARAMS CK_PTR CK_GCM_PARAMS_PTR
```

### 20.171.2.53 CK\_GOSTR3410\_DERIVE\_PARAMS

```
typedef struct CK_GOSTR3410_DERIVE_PARAMS CK_GOSTR3410_DERIVE_PARAMS
```

### 20.171.2.54 CK\_GOSTR3410\_DERIVE\_PARAMS\_PTR

```
typedef CK_GOSTR3410_DERIVE_PARAMS CK_PTR CK_GOSTR3410_DERIVE_PARAMS_PTR
```

### 20.171.2.55 CK\_GOSTR3410\_KEY\_WRAP\_PARAMS

```
typedef struct CK_GOSTR3410_KEY_WRAP_PARAMS CK_GOSTR3410_KEY_WRAP_PARAMS
```

### 20.171.2.56 CK\_GOSTR3410\_KEY\_WRAP\_PARAMS\_PTR

```
typedef CK_GOSTR3410_KEY_WRAP_PARAMS CK_PTR CK_GOSTR3410_KEY_WRAP_PARAMS_PTR
```

### 20.171.2.57 CK\_HW\_FEATURE\_TYPE

```
typedef CK_ULONG CK_HW_FEATURE_TYPE
```



**20.171.2.58 CK\_INFO**

```
typedef struct CK_INFO CK_INFO
```

**20.171.2.59 CK\_INFO\_PTR**

```
typedef CK_INFO CK_PTR CK_INFO_PTR
```

**20.171.2.60 CK\_JAVA\_MIDP\_SECURITY\_DOMAIN**

```
typedef CK_ULONG CK_JAVA_MIDP_SECURITY_DOMAIN
```

**20.171.2.61 CK\_KEA\_DERIVE\_PARAMS**

```
typedef struct CK_KEA_DERIVE_PARAMS CK_KEA_DERIVE_PARAMS
```

**20.171.2.62 CK\_KEA\_DERIVE\_PARAMS\_PTR**

```
typedef CK_KEA_DERIVE_PARAMS CK_PTR CK_KEA_DERIVE_PARAMS_PTR
```

**20.171.2.63 CK\_KEY\_DERIVATION\_STRING\_DATA**

```
typedef struct CK_KEY_DERIVATION_STRING_DATA CK_KEY_DERIVATION_STRING_DATA
```

**20.171.2.64 CK\_KEY\_DERIVATION\_STRING\_DATA\_PTR**

```
typedef CK_KEY_DERIVATION_STRING_DATA CK_PTR CK_KEY_DERIVATION_STRING_DATA_PTR
```

**20.171.2.65 CK\_KEY\_TYPE**

```
typedef CK_ULONG CK_KEY_TYPE
```

### 20.171.2.66 CK\_KEY\_WRAP\_SET\_OAEP\_PARAMS

```
typedef struct CK_KEY_WRAP_SET_OAEP_PARAMS CK_KEY_WRAP_SET_OAEP_PARAMS
```

### 20.171.2.67 CK\_KEY\_WRAP\_SET\_OAEP\_PARAMS\_PTR

```
typedef CK_KEY_WRAP_SET_OAEP_PARAMS CK_PTR CK_KEY_WRAP_SET_OAEP_PARAMS_PTR
```

### 20.171.2.68 CK\_KIP\_PARAMS

```
typedef struct CK_KIP_PARAMS CK_KIP_PARAMS
```

### 20.171.2.69 CK\_KIP\_PARAMS\_PTR

```
typedef CK_KIP_PARAMS CK_PTR CK_KIP_PARAMS_PTR
```

### 20.171.2.70 CK\_LONG

```
typedef long int CK_LONG
```

### 20.171.2.71 CK\_MAC\_GENERAL\_PARAMS

```
typedef CK_ULONG CK_MAC_GENERAL_PARAMS
```

### 20.171.2.72 CK\_MAC\_GENERAL\_PARAMS\_PTR

```
typedef CK_MAC_GENERAL_PARAMS CK_PTR CK_MAC_GENERAL_PARAMS_PTR
```

### 20.171.2.73 CK\_MECHANISM

```
typedef struct CK_MECHANISM CK_MECHANISM
```

**20.171.2.74 CK\_MECHANISM\_INFO**

```
typedef struct CK_MECHANISM_INFO CK_MECHANISM_INFO
```

**20.171.2.75 CK\_MECHANISM\_INFO\_PTR**

```
typedef CK_MECHANISM_INFO CK_PTR CK_MECHANISM_INFO_PTR
```

**20.171.2.76 CK\_MECHANISM\_PTR**

```
typedef CK_MECHANISM CK_PTR CK_MECHANISM_PTR
```

**20.171.2.77 CK\_MECHANISM\_TYPE**

```
typedef CK_ULONG CK_MECHANISM_TYPE
```

**20.171.2.78 CK\_MECHANISM\_TYPE\_PTR**

```
typedef CK_MECHANISM_TYPE CK_PTR CK_MECHANISM_TYPE_PTR
```

**20.171.2.79 CK\_NOTIFICATION**

```
typedef CK_ULONG CK_NOTIFICATION
```

**20.171.2.80 CK\_OBJECT\_CLASS**

```
typedef CK_ULONG CK_OBJECT_CLASS
```

**20.171.2.81 CK\_OBJECT\_CLASS\_PTR**

```
typedef CK_OBJECT_CLASS CK_PTR CK_OBJECT_CLASS_PTR
```

### 20.171.2.82 CK\_OBJECT\_HANDLE

```
typedef CK_ULONG CK_OBJECT_HANDLE
```

### 20.171.2.83 CK\_OBJECT\_HANDLE\_PTR

```
typedef CK_OBJECT_HANDLE CK_PTR CK_OBJECT_HANDLE_PTR
```

### 20.171.2.84 CK\_OTP\_PARAM

```
typedef struct CK_OTP_PARAM CK_OTP_PARAM
```

### 20.171.2.85 CK\_OTP\_PARAM\_PTR

```
typedef CK_OTP_PARAM CK_PTR CK_OTP_PARAM_PTR
```

### 20.171.2.86 CK\_OTP\_PARAM\_TYPE

```
typedef CK_ULONG CK_OTP_PARAM_TYPE
```

### 20.171.2.87 CK\_OTP\_PARAMS

```
typedef struct CK_OTP_PARAMS CK_OTP_PARAMS
```

### 20.171.2.88 CK\_OTP\_PARAMS\_PTR

```
typedef CK_OTP_PARAMS CK_PTR CK_OTP_PARAMS_PTR
```

### 20.171.2.89 CK\_OTP\_SIGNATURE\_INFO

```
typedef struct CK_OTP_SIGNATURE_INFO CK_OTP_SIGNATURE_INFO
```

**20.171.2.90 CK\_OTP\_SIGNATURE\_INFO\_PTR**

```
typedef CK_OTP_SIGNATURE_INFO CK_PTR CK_OTP_SIGNATURE_INFO_PTR
```

**20.171.2.91 CK\_PARAM\_TYPE**

```
typedef CK_OTP_PARAM_TYPE CK_PARAM_TYPE
```

**20.171.2.92 CK\_PBE\_PARAMS**

```
typedef struct CK_PBE_PARAMS CK_PBE_PARAMS
```

**20.171.2.93 CK\_PBE\_PARAMS\_PTR**

```
typedef CK_PBE_PARAMS CK_PTR CK_PBE_PARAMS_PTR
```

**20.171.2.94 CK\_PKCS5\_PBKD2\_PARAMS**

```
typedef struct CK_PKCS5_PBKD2_PARAMS CK_PKCS5_PBKD2_PARAMS
```

**20.171.2.95 CK\_PKCS5\_PBKD2\_PARAMS2**

```
typedef struct CK_PKCS5_PBKD2_PARAMS2 CK_PKCS5_PBKD2_PARAMS2
```

**20.171.2.96 CK\_PKCS5\_PBKD2\_PARAMS2\_PTR**

```
typedef CK_PKCS5_PBKD2_PARAMS2 CK_PTR CK_PKCS5_PBKD2_PARAMS2_PTR
```

**20.171.2.97 CK\_PKCS5\_PBKD2\_PARAMS\_PTR**

```
typedef CK_PKCS5_PBKD2_PARAMS CK_PTR CK_PKCS5_PBKD2_PARAMS_PTR
```

### 20.171.2.98 CK\_PKCS5\_PBKD2\_PSEUDO\_RANDOM\_FUNCTION\_TYPE

```
typedef CK_ULONG CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE
```

### 20.171.2.99 CK\_PKCS5\_PBKD2\_PSEUDO\_RANDOM\_FUNCTION\_TYPE\_PTR

```
typedef CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE CK_PTR CK_PKCS5_PBKD2_PSEUDO_RANDOM_FUNCTION_TYPE_PTR
```

### 20.171.2.100 CK\_PKCS5\_PBKDF2\_SALT\_SOURCE\_TYPE

```
typedef CK_ULONG CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE
```

### 20.171.2.101 CK\_PKCS5\_PBKDF2\_SALT\_SOURCE\_TYPE\_PTR

```
typedef CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE CK_PTR CK_PKCS5_PBKDF2_SALT_SOURCE_TYPE_PTR
```

### 20.171.2.102 CK\_RC2\_CBC\_PARAMS

```
typedef struct CK_RC2_CBC_PARAMS CK_RC2_CBC_PARAMS
```

### 20.171.2.103 CK\_RC2\_CBC\_PARAMS\_PTR

```
typedef CK_RC2_CBC_PARAMS CK_PTR CK_RC2_CBC_PARAMS_PTR
```

### 20.171.2.104 CK\_RC2\_MAC\_GENERAL\_PARAMS

```
typedef struct CK_RC2_MAC_GENERAL_PARAMS CK_RC2_MAC_GENERAL_PARAMS
```

### 20.171.2.105 CK\_RC2\_MAC\_GENERAL\_PARAMS\_PTR

```
typedef CK_RC2_MAC_GENERAL_PARAMS CK_PTR CK_RC2_MAC_GENERAL_PARAMS_PTR
```

**20.171.2.106 CK\_RC2\_PARAMS**

```
typedef CK_ULONG CK_RC2_PARAMS
```

**20.171.2.107 CK\_RC2\_PARAMS\_PTR**

```
typedef CK_RC2_PARAMS CK_PTR CK_RC2_PARAMS_PTR
```

**20.171.2.108 CK\_RC5\_CBC\_PARAMS**

```
typedef struct CK_RC5_CBC_PARAMS CK_RC5_CBC_PARAMS
```

**20.171.2.109 CK\_RC5\_CBC\_PARAMS\_PTR**

```
typedef CK_RC5_CBC_PARAMS CK_PTR CK_RC5_CBC_PARAMS_PTR
```

**20.171.2.110 CK\_RC5\_MAC\_GENERAL\_PARAMS**

```
typedef struct CK_RC5_MAC_GENERAL_PARAMS CK_RC5_MAC_GENERAL_PARAMS
```

**20.171.2.111 CK\_RC5\_MAC\_GENERAL\_PARAMS\_PTR**

```
typedef CK_RC5_MAC_GENERAL_PARAMS CK_PTR CK_RC5_MAC_GENERAL_PARAMS_PTR
```

**20.171.2.112 CK\_RC5\_PARAMS**

```
typedef struct CK_RC5_PARAMS CK_RC5_PARAMS
```

**20.171.2.113 CK\_RC5\_PARAMS\_PTR**

```
typedef CK_RC5_PARAMS CK_PTR CK_RC5_PARAMS_PTR
```

### 20.171.2.114 CK\_RSA\_AES\_KEY\_WRAP\_PARAMS

```
typedef struct CK_RSA_AES_KEY_WRAP_PARAMS CK_RSA_AES_KEY_WRAP_PARAMS
```

### 20.171.2.115 CK\_RSA\_AES\_KEY\_WRAP\_PARAMS\_PTR

```
typedef CK_RSA_AES_KEY_WRAP_PARAMS CK_PTR CK_RSA_AES_KEY_WRAP_PARAMS_PTR
```

### 20.171.2.116 CK\_RSA\_PKCS\_MGF\_TYPE

```
typedef CK_ULONG CK_RSA_PKCS_MGF_TYPE
```

### 20.171.2.117 CK\_RSA\_PKCS\_MGF\_TYPE\_PTR

```
typedef CK_RSA_PKCS_MGF_TYPE CK_PTR CK_RSA_PKCS_MGF_TYPE_PTR
```

### 20.171.2.118 CK\_RSA\_PKCS\_OAEP\_PARAMS

```
typedef struct CK_RSA_PKCS_OAEP_PARAMS CK_RSA_PKCS_OAEP_PARAMS
```

### 20.171.2.119 CK\_RSA\_PKCS\_OAEP\_PARAMS\_PTR

```
typedef CK_RSA_PKCS_OAEP_PARAMS CK_PTR CK_RSA_PKCS_OAEP_PARAMS_PTR
```

### 20.171.2.120 CK\_RSA\_PKCS\_OAEP\_SOURCE\_TYPE

```
typedef CK_ULONG CK_RSA_PKCS_OAEP_SOURCE_TYPE
```

### 20.171.2.121 CK\_RSA\_PKCS\_OAEP\_SOURCE\_TYPE\_PTR

```
typedef CK_RSA_PKCS_OAEP_SOURCE_TYPE CK_PTR CK_RSA_PKCS_OAEP_SOURCE_TYPE_PTR
```



**20.171.2.122 CK\_RSA\_PKCS\_PSS\_PARAMS**

```
typedef struct CK_RSA_PKCS_PSS_PARAMS CK_RSA_PKCS_PSS_PARAMS
```

**20.171.2.123 CK\_RSA\_PKCS\_PSS\_PARAMS\_PTR**

```
typedef CK_RSA_PKCS_PSS_PARAMS CK_PTR CK_RSA_PKCS_PSS_PARAMS_PTR
```

**20.171.2.124 CK\_RV**

```
typedef CK_ULONG CK_RV
```

**20.171.2.125 CK\_SEED\_CBC\_ENCRYPT\_DATA\_PARAMS**

```
typedef struct CK_SEED_CBC_ENCRYPT_DATA_PARAMS CK_SEED_CBC_ENCRYPT_DATA_PARAMS
```

**20.171.2.126 CK\_SEED\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR**

```
typedef CK_SEED_CBC_ENCRYPT_DATA_PARAMS CK_PTR CK_SEED_CBC_ENCRYPT_DATA_PARAMS_PTR
```

**20.171.2.127 CK\_SESSION\_HANDLE**

```
typedef CK_ULONG CK_SESSION_HANDLE
```

**20.171.2.128 CK\_SESSION\_HANDLE\_PTR**

```
typedef CK_SESSION_HANDLE CK_PTR CK_SESSION_HANDLE_PTR
```

**20.171.2.129 CK\_SESSION\_INFO**

```
typedef struct CK_SESSION_INFO CK_SESSION_INFO
```

### 20.171.2.130 CK\_SESSION\_INFO\_PTR

```
typedef CK_SESSION_INFO CK_PTR CK_SESSION_INFO_PTR
```

### 20.171.2.131 CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS

```
typedef struct CK_SKIPJACK_PRIVATE_WRAP_PARAMS CK_SKIPJACK_PRIVATE_WRAP_PARAMS
```

### 20.171.2.132 CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS\_PTR

```
typedef CK_SKIPJACK_PRIVATE_WRAP_PARAMS CK_PTR CK_SKIPJACK_PRIVATE_WRAP_PARAMS_PTR
```

### 20.171.2.133 CK\_SKIPJACK\_RELAYX\_PARAMS

```
typedef struct CK_SKIPJACK_RELAYX_PARAMS CK_SKIPJACK_RELAYX_PARAMS
```

### 20.171.2.134 CK\_SKIPJACK\_RELAYX\_PARAMS\_PTR

```
typedef CK_SKIPJACK_RELAYX_PARAMS CK_PTR CK_SKIPJACK_RELAYX_PARAMS_PTR
```

### 20.171.2.135 CK\_SLOT\_ID

```
typedef CK_ULONG CK_SLOT_ID
```

### 20.171.2.136 CK\_SLOT\_ID\_PTR

```
typedef CK_SLOT_ID CK_PTR CK_SLOT_ID_PTR
```

### 20.171.2.137 CK\_SLOT\_INFO

```
typedef struct CK_SLOT_INFO CK_SLOT_INFO
```

**20.171.2.138 CK\_SLOT\_INFO\_PTR**

```
typedef CK_SLOT_INFO CK_PTR CK_SLOT_INFO_PTR
```

**20.171.2.139 CK\_SSL3\_KEY\_MAT\_OUT**

```
typedef struct CK_SSL3_KEY_MAT_OUT CK_SSL3_KEY_MAT_OUT
```

**20.171.2.140 CK\_SSL3\_KEY\_MAT\_OUT\_PTR**

```
typedef CK_SSL3_KEY_MAT_OUT CK_PTR CK_SSL3_KEY_MAT_OUT_PTR
```

**20.171.2.141 CK\_SSL3\_KEY\_MAT\_PARAMS**

```
typedef struct CK_SSL3_KEY_MAT_PARAMS CK_SSL3_KEY_MAT_PARAMS
```

**20.171.2.142 CK\_SSL3\_KEY\_MAT\_PARAMS\_PTR**

```
typedef CK_SSL3_KEY_MAT_PARAMS CK_PTR CK_SSL3_KEY_MAT_PARAMS_PTR
```

**20.171.2.143 CK\_SSL3\_MASTER\_KEY\_DERIVE\_PARAMS**

```
typedef struct CK_SSL3_MASTER_KEY_DERIVE_PARAMS CK_SSL3_MASTER_KEY_DERIVE_PARAMS
```

**20.171.2.144 CK\_SSL3\_MASTER\_KEY\_DERIVE\_PARAMS\_PTR**

```
typedef struct CK_SSL3_MASTER_KEY_DERIVE_PARAMS CK_PTR CK_SSL3_MASTER_KEY_DERIVE_PARAMS_PTR
```

**20.171.2.145 CK\_SSL3\_RANDOM\_DATA**

```
typedef struct CK_SSL3_RANDOM_DATA CK_SSL3_RANDOM_DATA
```

### 20.171.2.146 CK\_STATE

```
typedef CK_ULONG CK_STATE
```

### 20.171.2.147 CK\_TLS12\_KEY\_MAT\_PARAMS

```
typedef struct CK_TLS12_KEY_MAT_PARAMS CK_TLS12_KEY_MAT_PARAMS
```

### 20.171.2.148 CK\_TLS12\_KEY\_MAT\_PARAMS\_PTR

```
typedef CK_TLS12_KEY_MAT_PARAMS CK_PTR CK_TLS12_KEY_MAT_PARAMS_PTR
```

### 20.171.2.149 CK\_TLS12\_MASTER\_KEY\_DERIVE\_PARAMS

```
typedef struct CK_TLS12_MASTER_KEY_DERIVE_PARAMS CK_TLS12_MASTER_KEY_DERIVE_PARAMS
```

### 20.171.2.150 CK\_TLS12\_MASTER\_KEY\_DERIVE\_PARAMS\_PTR

```
typedef CK_TLS12_MASTER_KEY_DERIVE_PARAMS CK_PTR CK_TLS12_MASTER_KEY_DERIVE_PARAMS_PTR
```

### 20.171.2.151 CK\_TLS\_KDF\_PARAMS

```
typedef struct CK_TLS_KDF_PARAMS CK_TLS_KDF_PARAMS
```

### 20.171.2.152 CK\_TLS\_KDF\_PARAMS\_PTR

```
typedef CK_TLS_KDF_PARAMS CK_PTR CK_TLS_KDF_PARAMS_PTR
```

### 20.171.2.153 CK\_TLS\_MAC\_PARAMS

```
typedef struct CK_TLS_MAC_PARAMS CK_TLS_MAC_PARAMS
```

**20.171.2.154 CK\_TLS\_MAC\_PARAMS\_PTR**

```
typedef CK_TLS_MAC_PARAMS CK_PTR CK_TLS_MAC_PARAMS_PTR
```

**20.171.2.155 CK\_TLS\_PRF\_PARAMS**

```
typedef struct CK_TLS_PRF_PARAMS CK_TLS_PRF_PARAMS
```

**20.171.2.156 CK\_TLS\_PRF\_PARAMS\_PTR**

```
typedef CK_TLS_PRF_PARAMS CK_PTR CK_TLS_PRF_PARAMS_PTR
```

**20.171.2.157 CK\_TOKEN\_INFO**

```
typedef struct CK_TOKEN_INFO CK_TOKEN_INFO
```

**20.171.2.158 CK\_TOKEN\_INFO\_PTR**

```
typedef CK_TOKEN_INFO CK_PTR CK_TOKEN_INFO_PTR
```

**20.171.2.159 CK\_ULONG**

```
typedef unsigned long int CK_ULONG
```

**20.171.2.160 CK\_ULONG\_PTR**

```
typedef CK_ULONG CK_PTR CK_ULONG_PTR
```

**20.171.2.161 CK\_USER\_TYPE**

```
typedef CK_ULONG CK_USER_TYPE
```

### 20.171.2.162 CK\_UTF8CHAR

```
typedef CK_BYTE CK_UTF8CHAR
```

### 20.171.2.163 CK\_UTF8CHAR\_PTR

```
typedef CK_UTF8CHAR CK_PTR CK_UTF8CHAR_PTR
```

### 20.171.2.164 CK\_VERSION

```
typedef struct CK_VERSION CK_VERSION
```

### 20.171.2.165 CK\_VERSION\_PTR

```
typedef CK_VERSION CK_PTR CK_VERSION_PTR
```

### 20.171.2.166 CK\_VOID\_PTR

```
typedef void CK_PTR CK_VOID_PTR
```

### 20.171.2.167 CK\_VOID\_PTR\_PTR

```
typedef CK_VOID_PTR CK_PTR CK_VOID_PTR_PTR
```

### 20.171.2.168 CK\_WTLS\_KEY\_MAT\_OUT

```
typedef struct CK_WTLS_KEY_MAT_OUT CK_WTLS_KEY_MAT_OUT
```

### 20.171.2.169 CK\_WTLS\_KEY\_MAT\_OUT\_PTR

```
typedef CK_WTLS_KEY_MAT_OUT CK_PTR CK_WTLS_KEY_MAT_OUT_PTR
```

**20.171.2.170 CK\_WTLS\_KEY\_MAT\_PARAMS**

```
typedef struct CK_WTLS_KEY_MAT_PARAMS CK_WTLS_KEY_MAT_PARAMS
```

**20.171.2.171 CK\_WTLS\_KEY\_MAT\_PARAMS\_PTR**

```
typedef CK_WTLS_KEY_MAT_PARAMS CK_PTR CK_WTLS_KEY_MAT_PARAMS_PTR
```

**20.171.2.172 CK\_WTLS\_MASTER\_KEY\_DERIVE\_PARAMS**

```
typedef struct CK_WTLS_MASTER_KEY_DERIVE_PARAMS CK_WTLS_MASTER_KEY_DERIVE_PARAMS
```

**20.171.2.173 CK\_WTLS\_MASTER\_KEY\_DERIVE\_PARAMS\_PTR**

```
typedef CK_WTLS_MASTER_KEY_DERIVE_PARAMS CK_PTR CK_WTLS_MASTER_KEY_DERIVE_PARAMS_PTR
```

**20.171.2.174 CK\_WTLS\_PRF\_PARAMS**

```
typedef struct CK_WTLS_PRF_PARAMS CK_WTLS_PRF_PARAMS
```

**20.171.2.175 CK\_WTLS\_PRF\_PARAMS\_PTR**

```
typedef CK_WTLS_PRF_PARAMS CK_PTR CK_WTLS_PRF_PARAMS_PTR
```

**20.171.2.176 CK\_WTLS\_RANDOM\_DATA**

```
typedef struct CK_WTLS_RANDOM_DATA CK_WTLS_RANDOM_DATA
```

**20.171.2.177 CK\_WTLS\_RANDOM\_DATA\_PTR**

```
typedef CK_WTLS_RANDOM_DATA CK_PTR CK_WTLS_RANDOM_DATA_PTR
```

### 20.171.2.178 CK\_X9\_42\_DH1\_DERIVE\_PARAMS

```
typedef struct CK_X9_42_DH1_DERIVE_PARAMS CK_X9_42_DH1_DERIVE_PARAMS
```

### 20.171.2.179 CK\_X9\_42\_DH1\_DERIVE\_PARAMS\_PTR

```
typedef struct CK_X9_42_DH1_DERIVE_PARAMS CK_PTR CK_X9_42_DH1_DERIVE_PARAMS_PTR
```

### 20.171.2.180 CK\_X9\_42\_DH2\_DERIVE\_PARAMS

```
typedef struct CK_X9_42_DH2_DERIVE_PARAMS CK_X9_42_DH2_DERIVE_PARAMS
```

### 20.171.2.181 CK\_X9\_42\_DH2\_DERIVE\_PARAMS\_PTR

```
typedef CK_X9_42_DH2_DERIVE_PARAMS CK_PTR CK_X9_42_DH2_DERIVE_PARAMS_PTR
```

### 20.171.2.182 CK\_X9\_42\_DH\_KDF\_TYPE

```
typedef CK_ULONG CK_X9_42_DH_KDF_TYPE
```

### 20.171.2.183 CK\_X9\_42\_DH\_KDF\_TYPE\_PTR

```
typedef CK_X9_42_DH_KDF_TYPE CK_PTR CK_X9_42_DH_KDF_TYPE_PTR
```

### 20.171.2.184 CK\_X9\_42\_MQV\_DERIVE\_PARAMS

```
typedef struct CK_X9_42_MQV_DERIVE_PARAMS CK_X9_42_MQV_DERIVE_PARAMS
```

### 20.171.2.185 CK\_X9\_42\_MQV\_DERIVE\_PARAMS\_PTR

```
typedef CK_X9_42_MQV_DERIVE_PARAMS CK_PTR CK_X9_42_MQV_DERIVE_PARAMS_PTR
```



**20.171.2.186 event**

```
typedef CK_NOTIFICATION event
```

**20.171.2.187 pApplication**

```
typedef CK_NOTIFICATION CK_VOID_PTR pApplication
```

**20.171.3 Function Documentation****20.171.3.1 CK\_CALLBACK\_FUNCTION() [1/5]**

```
typedef CK_CALLBACK_FUNCTION (
    CK_RV ,
    CK_CREATEMUTEX )
```

**20.171.3.2 CK\_CALLBACK\_FUNCTION() [2/5]**

```
typedef CK_CALLBACK_FUNCTION (
    CK_RV ,
    CK_DESTROYMUTEX )
```

**20.171.3.3 CK\_CALLBACK\_FUNCTION() [3/5]**

```
typedef CK_CALLBACK_FUNCTION (
    CK_RV ,
    CK_LOCKMUTEX )
```

**20.171.3.4 CK\_CALLBACK\_FUNCTION() [4/5]**

```
typedef CK_CALLBACK_FUNCTION (
    CK_RV ,
    CK_NOTIFY )
```

### 20.171.3.5 CK\_CALLBACK\_FUNCTION() [5/5]

```
typedef CK_CALLBACK_FUNCTION (
    CK_RV ,
    CK_UNLOCKMUTEX )
```

## 20.172 readme.md File Reference

## 20.173 README.md File Reference

## 20.174 README.md File Reference

## 20.175 README.md File Reference

## 20.176 README.md File Reference

## 20.177 README.md File Reference

## 20.178 README.md File Reference

## 20.179 README.md File Reference

## 20.180 README.md File Reference

## 20.181 README.md File Reference

## 20.182 README.md File Reference

## 20.183 secure\_boot.c File Reference

Provides required APIs to manage secure boot under various scenarios.

```
#include <string.h>
#include "secure_boot.h"
#include "io_protection_key.h"
#include "basic/atca_basic.h"
```

## Functions

- [ATCA\\_STATUS secure\\_boot\\_process](#) (void)  
*Handles secure boot functionality through initialization, execution, and de-initialization.*
- [ATCA\\_STATUS bind\\_host\\_and\\_secure\\_element\\_with\\_io\\_protection](#) (uint16\_t slot)  
*Binds host MCU and Secure element with IO protection key.*

### 20.183.1 Detailed Description

Provides required APIs to manage secure boot under various scenarios.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.183.2 Function Documentation

#### 20.183.2.1 bind\_host\_and\_secure\_element\_with\_io\_protection()

```
ATCA_STATUS bind_host_and_secure_element_with_io_protection (
    uint16_t slot )
```

Binds host MCU and Secure element with IO protection key.

##### Parameters

in	slot	The slot number of IO protection Key.
----	------	---------------------------------------

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

#### 20.183.2.2 secure\_boot\_process()

```
ATCA_STATUS secure_boot_process (
    void )
```

Handles secure boot functionality through initialization, execution, and de-initialization.

##### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.184 secure\_boot.h File Reference

Provides required APIs to manage secure boot under various scenarios.

```
#include "atca_status.h"
#include "secure_boot_memory.h"
#include "atca_command.h"
#include "crypto/atca_crypto_sw_sha2.h"
```

### Data Structures

- struct [secure\\_boot\\_config\\_bits](#)
- struct [secure\\_boot\\_parameters](#)

### Macros

- #define [SECURE\\_BOOT\\_CONFIG\\_DISABLE](#) 0
- #define [SECURE\\_BOOT\\_CONFIG\\_FULL\\_BOTH](#) 1
- #define [SECURE\\_BOOT\\_CONFIG\\_FULL\\_SIGN](#) 2
- #define [SECURE\\_BOOT\\_CONFIG\\_FULL\\_DIG](#) 3
- #define [SECURE\\_BOOT\\_CONFIGURATION](#) [SECURE\\_BOOT\\_CONFIG\\_FULL\\_DIG](#)
- #define [SECURE\\_BOOT\\_DIGEST\\_ENCRYPT\\_ENABLED](#) true
- #define [SECURE\\_BOOT\\_UPGRADE\\_SUPPORT](#) true

### Functions

- [ATCA\\_STATUS secure\\_boot\\_process](#) (void)  
*Handles secure boot functionality through initialization, execution, and de-initialization.*
- [ATCA\\_STATUS bind\\_host\\_and\\_secure\\_element\\_with\\_io\\_protection](#) (uint16\_t slot)  
*Binds host MCU and Secure element with IO protection key.*
- [ATCA\\_STATUS host\\_generate\\_random\\_number](#) (uint8\_t \*rand)

#### 20.184.1 Detailed Description

Provides required APIs to manage secure boot under various scenarios.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

#### 20.184.2 Macro Definition Documentation

#### 20.184.2.1 SECURE\_BOOT\_CONFIG\_DISABLE

```
#define SECURE_BOOT_CONFIG_DISABLE 0
```

#### 20.184.2.2 SECURE\_BOOT\_CONFIG\_FULL\_BOTH

```
#define SECURE_BOOT_CONFIG_FULL_BOTH 1
```

#### 20.184.2.3 SECURE\_BOOT\_CONFIG\_FULL\_DIG

```
#define SECURE_BOOT_CONFIG_FULL_DIG 3
```

#### 20.184.2.4 SECURE\_BOOT\_CONFIG\_FULL\_SIGN

```
#define SECURE_BOOT_CONFIG_FULL_SIGN 2
```

#### 20.184.2.5 SECURE\_BOOT\_CONFIGURATION

```
#define SECURE_BOOT_CONFIGURATION SECURE\_BOOT\_CONFIG\_FULL\_DIG
```

#### 20.184.2.6 SECURE\_BOOT\_DIGEST\_ENCRYPT\_ENABLED

```
#define SECURE_BOOT_DIGEST_ENCRYPT_ENABLED true
```

#### 20.184.2.7 SECURE\_BOOT\_UPGRADE\_SUPPORT

```
#define SECURE_BOOT_UPGRADE_SUPPORT true
```

### 20.184.3 Function Documentation

#### 20.184.3.1 bind\_host\_and\_secure\_element\_with\_io\_protection()

```
ATCA\_STATUS bind_host_and_secure_element_with_io_protection (
    uint16_t slot )
```

Binds host MCU and Secure element with IO protection key.

## 20.185 secure\_boot\_memory.h File Reference

---

### Parameters

<code>in</code>	<code>slot</code>	The slot number of IO protection Key.
-----------------	-------------------	---------------------------------------

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

### 20.184.3.2 host\_generate\_random\_number()

```
ATCA_STATUS host_generate_random_number (
    uint8_t * rand )
```

### 20.184.3.3 secure\_boot\_process()

```
ATCA_STATUS secure_boot_process (
    void )
```

Handles secure boot functionality through initialization, execution, and de-initialization.

### Returns

ATCA\_SUCCESS on success, otherwise an error code.

## 20.185 secure\_boot\_memory.h File Reference

Provides interface to memory component for the secure boot.

```
#include "atca_status.h"
#include "atca_command.h"
```

### Data Structures

- struct [memory\\_parameters](#)

### Functions

- [ATCA\\_STATUS secure\\_boot\\_init\\_memory](#) ([memory\\_parameters](#) \*memory\_params)
- [ATCA\\_STATUS secure\\_boot\\_read\\_memory](#) (uint8\_t \*pu8\_data, uint32\_t \*pu32\_target\_length)
- [ATCA\\_STATUS secure\\_boot\\_write\\_memory](#) (uint8\_t \*pu8\_data, uint32\_t \*pu32\_target\_length)
- void [secure\\_boot\\_deinit\\_memory](#) ([memory\\_parameters](#) \*memory\_params)
- [ATCA\\_STATUS secure\\_boot\\_mark\\_full\\_copy\\_completion](#) (void)
- bool [secure\\_boot\\_check\\_full\\_copy\\_completion](#) (void)

## 20.185.1 Detailed Description

Provides interface to memory component for the secure boot.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.185.2 Function Documentation

### 20.185.2.1 `secure_boot_check_full_copy_completion()`

```
bool secure_boot_check_full_copy_completion (
    void )
```

### 20.185.2.2 `secure_boot_deinit_memory()`

```
void secure_boot_deinit_memory (
    memory_parameters * memory_params )
```

### 20.185.2.3 `secure_boot_init_memory()`

```
ATCA_STATUS secure_boot_init_memory (
    memory_parameters * memory_params )
```

### 20.185.2.4 `secure_boot_mark_full_copy_completion()`

```
ATCA_STATUS secure_boot_mark_full_copy_completion (
    void )
```

### 20.185.2.5 `secure_boot_read_memory()`

```
ATCA_STATUS secure_boot_read_memory (
    uint8_t * pu8_data,
    uint32_t * pu32_target_length )
```

### 20.185.2.6 secure\_boot\_write\_memory()

```
ATCA_STATUS secure_boot_write_memory (
    uint8_t * pu8_data,
    uint32_t * pu32_target_length )
```

## 20.186 sha1\_routines.c File Reference

Software implementation of the SHA1 algorithm.

```
#include "sha1_routines.h"
#include <string.h>
#include "atca_compiler.h"
```

### Functions

- void [CL\\_hashInit](#) ([CL\\_HashContext](#) \*ctx)  
*Initialize context for performing SHA1 hash in software.*
- void [CL\\_hashUpdate](#) ([CL\\_HashContext](#) \*ctx, const uint8\_t \*src, int nbytes)  
*Add arbitrary data to a SHA1 hash.*
- void [CL\\_hashFinal](#) ([CL\\_HashContext](#) \*ctx, uint8\_t \*dest)  
*Complete the SHA1 hash in software and return the digest.*
- void [CL\\_hash](#) (uint8\_t \*msg, int msgBytes, uint8\_t \*dest)  
*Perform SHA1 hash of data in software.*
- void [shaEngine](#) (uint32\_t \*buf, uint32\_t \*h)

### 20.186.1 Detailed Description

Software implementation of the SHA1 algorithm.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.186.2 Function Documentation

#### 20.186.2.1 CL\_hash()

```
void CL_hash (
    uint8_t * msg,
    int msgBytes,
    uint8_t * dest )
```

Perform SHA1 hash of data in software.



## Parameters

in	<i>msg</i>	Data to be hashed
in	<i>msgBytes</i>	Data size in bytes
out	<i>dest</i>	Digest is returned here (20 bytes)

**20.186.2.2 CL\_hashFinal()**

```
void CL_hashFinal (
    CL_HashContext * ctx,
    uint8_t * dest )
```

Complete the SHA1 hash in software and return the digest.

## Parameters

in	<i>ctx</i>	Hash context
out	<i>dest</i>	Digest is returned here (20 bytes)

**20.186.2.3 CL\_hashInit()**

```
void CL_hashInit (
    CL_HashContext * ctx )
```

Initialize context for performing SHA1 hash in software.

## Parameters

in	<i>ctx</i>	Hash context
----	------------	--------------

**20.186.2.4 CL\_hashUpdate()**

```
void CL_hashUpdate (
    CL_HashContext * ctx,
    const uint8_t * src,
    int nbytes )
```

Add arbitrary data to a SHA1 hash.

## Parameters

in	<i>ctx</i>	Hash context
in	<i>src</i>	Data to be added to the hash
in	<i>nbytes</i>	Data size in bytes

### 20.186.2.5 shaEngine()

```
void shaEngine (
    uint32_t * buf,
    uint32_t * h )
```

## 20.187 sha1\_routines.h File Reference

Software implementation of the SHA1 algorithm.

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <stdint.h>
```

### Data Structures

- struct [CL\\_HashContext](#)

### Macros

- #define [U8](#) uint8\_t
- #define [U16](#) uint16\_t
- #define [U32](#) uint32\_t
- #define [memcpy\\_P](#) memmove
- #define [strcpy\\_P](#) strcpy
- #define [\\_WDRESET](#)()
- #define [\\_NOP](#)()
- #define [leftRotate](#)(x, n) (x) = (((x) << (n)) | ((x) >> (32 - (n))))

### Functions

- void [shaEngine](#) (uint32\_t \*buf, uint32\_t \*h)
- void [CL\\_hashInit](#) ([CL\\_HashContext](#) \*ctx)  
*Initialize context for performing SHA1 hash in software.*
- void [CL\\_hashUpdate](#) ([CL\\_HashContext](#) \*ctx, const uint8\_t \*src, int nbytes)  
*Add arbitrary data to a SHA1 hash.*
- void [CL\\_hashFinal](#) ([CL\\_HashContext](#) \*ctx, uint8\_t \*dest)  
*Complete the SHA1 hash in software and return the digest.*
- void [CL\\_hash](#) (uint8\_t \*msg, int msgBytes, uint8\_t \*dest)  
*Perform SHA1 hash of data in software.*

### 20.187.1 Detailed Description

Software implementation of the SHA1 algorithm.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.187.2 Macro Definition Documentation

#### 20.187.2.1 \_NOP

```
#define _NOP( )
```

#### 20.187.2.2 \_WDRESET

```
#define _WDRESET( )
```

#### 20.187.2.3 leftRotate

```
#define leftRotate(  
    x,  
    n ) (x) = (((x) << (n)) | ((x) >> (32 - (n))))
```

#### 20.187.2.4 memcpy\_P

```
#define memcpy_P memmove
```

#### 20.187.2.5 strcpy\_P

```
#define strcpy_P strcpy
```

### 20.187.2.6 U16

```
#define U16 uint16_t
```

### 20.187.2.7 U32

```
#define U32 uint32_t
```

### 20.187.2.8 U8

```
#define U8 uint8_t
```

## 20.187.3 Function Documentation

### 20.187.3.1 CL\_hash()

```
void CL_hash (
    uint8_t * msg,
    int msgBytes,
    uint8_t * dest )
```

Perform SHA1 hash of data in software.

#### Parameters

in	<i>msg</i>	Data to be hashed
in	<i>msgBytes</i>	Data size in bytes
out	<i>dest</i>	Digest is returned here (20 bytes)

### 20.187.3.2 CL\_hashFinal()

```
void CL_hashFinal (
    CL_HashContext * ctx,
    uint8_t * dest )
```

Complete the SHA1 hash in software and return the digest.

## Parameters

in	<i>ctx</i>	Hash context
out	<i>dest</i>	Digest is returned here (20 bytes)

**20.187.3.3 CL\_hashInit()**

```
void CL_hashInit (
    CL_HashContext * ctx )
```

Initialize context for performing SHA1 hash in software.

## Parameters

in	<i>ctx</i>	Hash context
----	------------	--------------

**20.187.3.4 CL\_hashUpdate()**

```
void CL_hashUpdate (
    CL_HashContext * ctx,
    const uint8_t * src,
    int nbytes )
```

Add arbitrary data to a SHA1 hash.

## Parameters

in	<i>ctx</i>	Hash context
in	<i>src</i>	Data to be added to the hash
in	<i>nbytes</i>	Data size in bytes

**20.187.3.5 shaEngine()**

```
void shaEngine (
    uint32_t * buf,
    uint32_t * h )
```

**20.188 sha2\_routines.c File Reference**

Software implementation of the SHA256 algorithm.

```
#include <string.h>
#include "sha2_routines.h"
#include "atca_compiler.h"
```

### Macros

- #define `rotate_right`(value, places) ((value >> places) | (value << (32 - places)))

### Functions

- void `sw_sha256_init` (`sw_sha256_ctx` \*ctx)  
*Intialize the software SHA256.*
- void `sw_sha256_update` (`sw_sha256_ctx` \*ctx, const uint8\_t \*msg, uint32\_t msg\_size)  
*updates the running hash with the next block of data, called iteratively for the entire stream of data to be hashed using the SHA256 software*
- void `sw_sha256_final` (`sw_sha256_ctx` \*ctx, uint8\_t digest[(32)])  
*completes the final SHA256 calculation and returns the final digest/hash*
- void `sw_sha256` (const uint8\_t \*message, unsigned int len, uint8\_t digest[(32)])  
*single call convenience function which computes Hash of given data using SHA256 software*

### 20.188.1 Detailed Description

Software implementation of the SHA256 algorithm.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.188.2 Macro Definition Documentation

#### 20.188.2.1 rotate\_right

```
#define rotate_right(  
    value,  
    places ) ((value >> places) | (value << (32 - places)))
```

### 20.188.3 Function Documentation

#### 20.188.3.1 sw\_sha256()

```
void sw_sha256 (  
    const uint8_t * message,  
    unsigned int len,  
    uint8_t digest[(32)] )
```

single call convenience function which computes Hash of given data using SHA256 software

## Parameters

in	<i>message</i>	pointer to stream of data to hash
in	<i>len</i>	size of data stream to hash
out	<i>digest</i>	result

**20.188.3.2 sw\_sha256\_final()**

```
void sw_sha256_final (
    sw_sha256_ctx * ctx,
    uint8_t digest[(32)] )
```

completes the final SHA256 calculation and returns the final digest/hash

## Parameters

in	<i>ctx</i>	ptr to context data structure
out	<i>digest</i>	receives the computed digest of the SHA 256

**20.188.3.3 sw\_sha256\_init()**

```
void sw_sha256_init (
    sw_sha256_ctx * ctx )
```

Intialize the software SHA256.

## Parameters

in	<i>ctx</i>	SHA256 hash context
----	------------	---------------------

**20.188.3.4 sw\_sha256\_update()**

```
void sw_sha256_update (
    sw_sha256_ctx * ctx,
    const uint8_t * msg,
    uint32_t msg_size )
```

updates the running hash with the next block of data, called iteratively for the entire stream of data to be hashed using the SHA256 software

### Parameters

in	<i>ctx</i>	SHA256 hash context
in	<i>msg</i>	Raw blocks to be processed
in	<i>msg_size</i>	The size of the message passed

## 20.189 sha2\_routines.h File Reference

Software implementation of the SHA256 algorithm.

```
#include <stdint.h>
```

### Data Structures

- struct [sw\\_sha256\\_ctx](#)

### Macros

- #define [SHA256\\_DIGEST\\_SIZE](#) (32)
- #define [SHA256\\_BLOCK\\_SIZE](#) (64)

### Functions

- void [sw\\_sha256\\_init](#) ([sw\\_sha256\\_ctx](#) \*ctx)  
*Intialize the software SHA256.*
- void [sw\\_sha256\\_update](#) ([sw\\_sha256\\_ctx](#) \*ctx, const uint8\_t \*message, uint32\_t len)  
*updates the running hash with the next block of data, called iteratively for the entire stream of data to be hashed using the SHA256 software*
- void [sw\\_sha256\\_final](#) ([sw\\_sha256\\_ctx](#) \*ctx, uint8\_t digest[(32)])  
*completes the final SHA256 calculation and returns the final digest/hash*
- void [sw\\_sha256](#) (const uint8\_t \*message, unsigned int len, uint8\_t digest[(32)])  
*single call convenience function which computes Hash of given data using SHA256 software*

### 20.189.1 Detailed Description

Software implementation of the SHA256 algorithm.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.189.2 Macro Definition Documentation



### 20.189.2.1 SHA256\_BLOCK\_SIZE

```
#define SHA256_BLOCK_SIZE (64)
```

### 20.189.2.2 SHA256\_DIGEST\_SIZE

```
#define SHA256_DIGEST_SIZE (32)
```

## 20.189.3 Function Documentation

### 20.189.3.1 sw\_sha256()

```
void sw_sha256 (
    const uint8_t * message,
    unsigned int len,
    uint8_t digest[(32)] )
```

single call convenience function which computes Hash of given data using SHA256 software

#### Parameters

in	<i>message</i>	pointer to stream of data to hash
in	<i>len</i>	size of data stream to hash
out	<i>digest</i>	result

### 20.189.3.2 sw\_sha256\_final()

```
void sw_sha256_final (
    sw_sha256_ctx * ctx,
    uint8_t digest[(32)] )
```

completes the final SHA256 calculation and returns the final digest/hash

#### Parameters

in	<i>ctx</i>	ptr to context data structure
out	<i>digest</i>	receives the computed digest of the SHA 256

### 20.189.3.3 sw\_sha256\_init()

```
void sw_sha256_init (
    sw_sha256_ctx * ctx )
```

Intialize the software SHA256.

#### Parameters

in	ctx	SHA256 hash context
----	-----	---------------------

### 20.189.3.4 sw\_sha256\_update()

```
void sw_sha256_update (
    sw_sha256_ctx * ctx,
    const uint8_t * msg,
    uint32_t msg_size )
```

updates the running hash with the next block of data, called iteratively for the entire stream of data to be hashed using the SHA256 software

#### Parameters

in	ctx	SHA256 hash context
in	msg	Raw blocks to be processed
in	msg_size	The size of the message passed

## 20.190 swi\_uart\_samd21\_asf.c File Reference

ATXMEGA's ATCA Hardware abstraction layer for SWI interface over UART drivers.

```
#include <stdlib.h>
#include <stdio.h>
#include "swi_uart_samd21_asf.h"
#include "atca_helpers.h"
```

## Functions

- [ATCA\\_STATUS swi\\_uart\\_init](#) ([ATCASWIMaster\\_t](#) \*instance)  
*Implementation of SWI UART init.*
- [ATCA\\_STATUS swi\\_uart\\_deinit](#) ([ATCASWIMaster\\_t](#) \*instance)  
*Implementation of SWI UART deinit.*
- void [swi\\_uart\\_setbaud](#) ([ATCASWIMaster\\_t](#) \*instance, uint32\_t baudrate)  
*implementation of SWI UART change baudrate.*
- void [swi\\_uart\\_mode](#) ([ATCASWIMaster\\_t](#) \*instance, uint8\_t mode)

*implementation of SWI UART change mode.*

- void [swi\\_uart\\_discover\\_buses](#) (int swi\_uart\_buses[], int max\_buses)  
*discover UART buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge*
- [ATCA\\_STATUS swi\\_uart\\_send\\_byte](#) ([ATCASWIMaster\\_t](#) \*instance, uint8\_t data)  
*HAL implementation of SWI UART send byte over ASF. This function send one byte over UART.*
- [ATCA\\_STATUS swi\\_uart\\_receive\\_byte](#) ([ATCASWIMaster\\_t](#) \*instance, uint8\_t \*data)  
*HAL implementation of SWI UART receive bytes over ASF. This function receive one byte over UART.*

## Variables

- struct port\_config [pin\\_conf](#)

### 20.190.1 Detailed Description

ATXMEGA's ATCA Hardware abstraction layer for SWI interface over UART drivers.

Prerequisite: add UART Polled support to application in Atmel Studio

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.191 swi\_uart\_samd21\_asf.h File Reference

ATXMEGA's ATCA Hardware abstraction layer for SWI interface over UART drivers.

```
#include <asf.h>
#include "cryptoauthlib.h"
```

## Data Structures

- struct [atcaSWImaster](#)  
*this is the hal\_data for ATCA HAL for ASF SERCOM*

## Macros

- #define [MAX\\_SWI\\_BUSES](#) 6
- #define [RECEIVE\\_MODE](#) 0
- #define [TRANSMIT\\_MODE](#) 1
- #define [RX\\_DELAY](#) 10
- #define [TX\\_DELAY](#) 90
- #define [DEBUG\\_PIN\\_1](#) EXT2\_PIN\_5
- #define [DEBUG\\_PIN\\_2](#) EXT2\_PIN\_6

## Typedefs

- typedef struct [atcaSWImaster](#) [ATCASWIMaster\\_t](#)  
*this is the hal\_data for ATCA HAL for ASF SERCOM*

## Functions

- [ATCA\\_STATUS swi\\_uart\\_init](#) ([ATCASWIMaster\\_t](#) \*instance)  
*Implementation of SWI UART init.*
- [ATCA\\_STATUS swi\\_uart\\_deinit](#) ([ATCASWIMaster\\_t](#) \*instance)  
*Implementation of SWI UART deinit.*
- void [swi\\_uart\\_setbaud](#) ([ATCASWIMaster\\_t](#) \*instance, uint32\_t baudrate)  
*implementation of SWI UART change baudrate.*
- void [swi\\_uart\\_mode](#) ([ATCASWIMaster\\_t](#) \*instance, uint8\_t mode)  
*implementation of SWI UART change mode.*
- void [swi\\_uart\\_discover\\_buses](#) (int swi\_uart\_buses[], int max\_buses)  
*discover UART buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge*
- [ATCA\\_STATUS swi\\_uart\\_send\\_byte](#) ([ATCASWIMaster\\_t](#) \*instance, uint8\_t data)  
*HAL implementation of SWI UART send byte over ASF. This function send one byte over UART.*
- [ATCA\\_STATUS swi\\_uart\\_receive\\_byte](#) ([ATCASWIMaster\\_t](#) \*instance, uint8\_t \*data)  
*HAL implementation of SWI UART receive bytes over ASF. This function receive one byte over UART.*

### 20.191.1 Detailed Description

ATXMEGA's ATCA Hardware abstraction layer for SWI interface over UART drivers.

Prerequisite: add UART Polled support to application in Atmel Studio

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.192 swi\_uart\_start.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <peripheral_clk_config.h>
#include "swi_uart_start.h"
#include "atca_helpers.h"
```

## Macros

- #define [USART\\_BAUD\\_RATE](#)(baud, sercom\_freq) (65536 - ((65536 \* 16.0F \* baud) / sercom\_freq))

## Functions

- [ATCA\\_STATUS swi\\_uart\\_init](#) ([ATCASWIMaster\\_t](#) \*instance)  
*Implementation of SWI UART init.*
- [ATCA\\_STATUS swi\\_uart\\_deinit](#) ([ATCASWIMaster\\_t](#) \*instance)  
*Implementation of SWI UART deinit.*
- void [swi\\_uart\\_setbaud](#) ([ATCASWIMaster\\_t](#) \*instance, uint32\_t baudrate)  
*implementation of SWI UART change baudrate.*
- void [swi\\_uart\\_mode](#) ([ATCASWIMaster\\_t](#) \*instance, uint8\_t mode)  
*implementation of SWI UART change mode.*
- void [swi\\_uart\\_discover\\_buses](#) (int swi\_uart\_buses[], int max\_buses)  
*discover UART buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge*
- [ATCA\\_STATUS swi\\_uart\\_send\\_byte](#) ([ATCASWIMaster\\_t](#) \*instance, uint8\_t data)  
*HAL implementation of SWI UART send byte over ASF. This function send one byte over UART.*
- [ATCA\\_STATUS swi\\_uart\\_receive\\_byte](#) ([ATCASWIMaster\\_t](#) \*instance, uint8\_t \*data)  
*HAL implementation of SWI UART receive bytes over ASF. This function receive one byte over UART.*

### 20.192.1 Detailed Description

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.192.2 Macro Definition Documentation

#### 20.192.2.1 USART\_BAUD\_RATE

```
#define USART_BAUD_RATE(  
    baud,  
    sercom_freq ) (65536 - ((65536 * 16.0F * baud) / sercom_freq))
```

## 20.193 swi\_uart\_start.h File Reference

```
#include <stdlib.h>  
#include "atmel_start.h"  
#include "cryptoauthlib.h"
```

## Data Structures

- struct [atcaSWImaster](#)  
*this is the hal\_data for ATCA HAL for ASF SERCOM*

## Macros

- #define [MAX\\_SWI\\_BUSES](#) 6
- #define [RECEIVE\\_MODE](#) 0
- #define [TRANSMIT\\_MODE](#) 1
- #define [RX\\_DELAY](#) 10
- #define [TX\\_DELAY](#) 93

## Typedefs

- typedef struct [atcaSWIMaster](#) [ATCASWIMaster\\_t](#)  
*this is the hal\_data for ATCA HAL for ASF SERCOM*

## Functions

- [ATCA\\_STATUS swi\\_uart\\_init](#) ([ATCASWIMaster\\_t](#) \*instance)  
*Implementation of SWI UART init.*
- [ATCA\\_STATUS swi\\_uart\\_deinit](#) ([ATCASWIMaster\\_t](#) \*instance)  
*Implementation of SWI UART deinit.*
- void [swi\\_uart\\_setbaud](#) ([ATCASWIMaster\\_t](#) \*instance, uint32\_t baudrate)  
*implementation of SWI UART change baudrate.*
- void [swi\\_uart\\_mode](#) ([ATCASWIMaster\\_t](#) \*instance, uint8\_t mode)  
*implementation of SWI UART change mode.*
- void [swi\\_uart\\_discover\\_buses](#) (int swi\_uart\_buses[], int max\_buses)  
*discover UART buses available for this hardware this maintains a list of logical to physical bus mappings freeing the application of the a-priori knowledge*
- [ATCA\\_STATUS swi\\_uart\\_send\\_byte](#) ([ATCASWIMaster\\_t](#) \*instance, uint8\_t data)  
*HAL implementation of SWI UART send byte over ASF. This function send one byte over UART.*
- [ATCA\\_STATUS swi\\_uart\\_receive\\_byte](#) ([ATCASWIMaster\\_t](#) \*instance, uint8\_t \*data)  
*HAL implementation of SWI UART receive bytes over ASF. This function receive one byte over UART.*

### 20.193.1 Detailed Description

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.194 symmetric\_authentication.c File Reference

Contains API for performing the symmetric Authentication between the Host and the device.

```
#include "cryptoauthlib.h"
#include "host/atca_host.h"
#include "symmetric_authentication.h"
```

## Functions

- [ATCA\\_STATUS symmetric\\_authenticate](#) (uint8\_t slot, const uint8\_t \*master\_key, const uint8\_t \*rand\_↔ number)

*Function which does the authentication between the host and device.*

### 20.194.1 Detailed Description

Contains API for performing the symmetric Authentication between the Host and the device.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.194.2 Function Documentation

#### 20.194.2.1 symmetric\_authenticate()

```
ATCA_STATUS symmetric_authenticate (
    uint8_t slot,
    const uint8_t * master_key,
    const uint8_t * rand_number )
```

Function which does the authentication between the host and device.

#### Parameters

in	<i>slot</i>	The slot number used for the symmetric authentication.
in	<i>master_key</i>	The master key used for the calculating the symmetric key.
in	<i>rand_number</i>	The 20 byte rand_number from the host.

#### Returns

ATCA\_SUCCESS on successful authentication, otherwise an error code.

## 20.195 symmetric\_authentication.h File Reference

Contains API for performing the symmetric Authentication between the Host and the device.

```
#include "cryptoauthlib.h"
```

## Functions

- [ATCA\\_STATUS symmetric\\_authenticate](#) (uint8\_t slot, const uint8\_t \*master\_key, const uint8\_t \*rand\_↔ number)

*Function which does the authentication between the host and device.*

## 20.195.1 Detailed Description

Contains API for performing the symmetric Authentication between the Host and the device.

### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.195.2 Function Documentation

### 20.195.2.1 symmetric\_authenticate()

```
ATCA_STATUS symmetric_authenticate (
    uint8_t slot,
    const uint8_t * master_key,
    const uint8_t * rand_number )
```

Function which does the authentication between the host and device.

#### Parameters

in	<i>slot</i>	The slot number used for the symmetric authentication.
in	<i>master_key</i>	The master key used for the calculating the symmetric key.
in	<i>rand_number</i>	The 20 byte rand_number from the host.

#### Returns

ATCA\_SUCCESS on successful authentication, otherwise an error code.

## 20.196 tflxtls\_cert\_def\_4\_device.c File Reference

TNG TLS device certificate definition.

```
#include "atcacert/atcacert_def.h"
#include "tngtls_cert_def_1_signer.h"
```

### Variables

- const uint8\_t [g\\_tflxtls\\_cert\\_template\\_4\\_device](#) [500]
- const [atcacert\\_cert\\_element\\_t](#) [g\\_tflxtls\\_cert\\_elements\\_4\\_device](#) []
- const [atcacert\\_def\\_t](#) [g\\_tflxtls\\_cert\\_def\\_4\\_device](#)



### 20.196.1 Detailed Description

TNG TLS device certificate definition.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.196.2 Variable Documentation

#### 20.196.2.1 g\_tflxtls\_cert\_elements\_4\_device

```
const atcacert_cert_element_t g_tflxtls_cert_elements_4_device[ ]
```

#### 20.196.2.2 g\_tflxtls\_cert\_template\_4\_device

```
const uint8_t g_tflxtls_cert_template_4_device[500]
```

## 20.197 tflxtls\_cert\_def\_4\_device.h File Reference

TNG TLS device certificate definition.

```
#include "atcacert/atcacert_def.h"
```

### Variables

- const atcacert\_def\_t g\_tflxtls\_cert\_def\_4\_device

### 20.197.1 Detailed Description

TNG TLS device certificate definition.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.198 tng\_atca.c File Reference

TNG Helper Functions.

```
#include <string.h>
#include "cryptoauthlib.h"
#include "tng_atca.h"
#include "tnglora_cert_def_2_device.h"
#include "tnglora_cert_def_4_device.h"
#include "tngtls_cert_def_2_device.h"
#include "tngtls_cert_def_3_device.h"
#include "tflxtls_cert_def_4_device.h"
#include "atcacert/atcacert_def.h"
```

### Data Structures

- struct [tng\\_cert\\_map\\_element](#)

### Functions

- const [atcacert\\_def\\_t](#) \* [tng\\_map\\_get\\_device\\_cert\\_def](#) (int index)  
*Helper function to iterate through all trust cert definitions.*
- [ATCA\\_STATUS](#) [tng\\_get\\_device\\_cert\\_def](#) (const [atcacert\\_def\\_t](#) \*\*cert\_def)  
*Get the TNG device certificate definition.*
- [ATCA\\_STATUS](#) [tng\\_get\\_device\\_pubkey](#) (uint8\_t \*public\_key)  
*Uses GenKey command to calculate the public key from the primary device public key.*

### 20.198.1 Detailed Description

TNG Helper Functions.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.199 tng\_atca.h File Reference

TNG Helper Functions.

```
#include "atca_basic.h"
#include "atcacert/atcacert_def.h"
```

## Functions

- const [atcacert\\_def\\_t](#) \* [tng\\_map\\_get\\_device\\_cert\\_def](#) (int index)  
*Helper function to iterate through all trust cert definitions.*
- [ATCA\\_STATUS](#) [tng\\_get\\_device\\_cert\\_def](#) (const [atcacert\\_def\\_t](#) \*\*cert\_def)  
*Get the TNG device certificate definition.*
- [ATCA\\_STATUS](#) [tng\\_get\\_device\\_pubkey](#) (uint8\_t \*public\_key)  
*Uses GenKey command to calculate the public key from the primary device public key.*

### 20.199.1 Detailed Description

TNG Helper Functions.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.200 tng\_atcacert\_client.c File Reference

Client side certificate I/O functions for TNG devices.

```
#include "tng_atca.h"
#include "atcacert/atcacert_client.h"
#include "tng_atcacert_client.h"
#include "tngtls_cert_def_1_signer.h"
#include "tng_root_cert.h"
```

## Functions

- int [tng\\_atcacert\\_max\\_device\\_cert\\_size](#) (size\_t \*max\_cert\_size)  
*Return the maximum possible certificate size in bytes for a TNG device certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.*
- int [tng\\_atcacert\\_read\\_device\\_cert](#) (uint8\_t \*cert, size\_t \*cert\_size, const uint8\_t \*signer\_cert)  
*Reads the device certificate for a TNG device.*
- int [tng\\_atcacert\\_device\\_public\\_key](#) (uint8\_t \*public\_key, uint8\_t \*cert)  
*Reads the device public key.*
- int [tng\\_atcacert\\_max\\_signer\\_cert\\_size](#) (size\_t \*max\_cert\_size)  
*Return the maximum possible certificate size in bytes for a TNG signer certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.*
- int [tng\\_atcacert\\_read\\_signer\\_cert](#) (uint8\_t \*cert, size\_t \*cert\_size)  
*Reads the signer certificate for a TNG device.*
- int [tng\\_atcacert\\_signer\\_public\\_key](#) (uint8\_t \*public\_key, uint8\_t \*cert)  
*Reads the signer public key.*
- int [tng\\_atcacert\\_root\\_cert\\_size](#) (size\_t \*cert\_size)  
*Get the size of the TNG root cert.*
- int [tng\\_atcacert\\_root\\_cert](#) (uint8\_t \*cert, size\_t \*cert\_size)  
*Get the TNG root cert.*
- int [tng\\_atcacert\\_root\\_public\\_key](#) (uint8\_t \*public\_key)  
*Gets the root public key.*

### 20.200.1 Detailed Description

Client side certificate I/O functions for TNG devices.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.200.2 Function Documentation

#### 20.200.2.1 tng\_atcacert\_device\_public\_key()

```
int tng_atcacert_device_public_key (
    uint8_t * public_key,
    uint8_t * cert )
```

Reads the device public key.

#### Parameters

out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve.
in	<i>cert</i>	If supplied, the device public key is used from this certificate. If set to NULL, the device public key is read from the device.

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

#### 20.200.2.2 tng\_atcacert\_max\_signer\_cert\_size()

```
int tng_atcacert_max_signer_cert_size (
    size_t * max_cert_size )
```

Return the maximum possible certificate size in bytes for a TNG signer certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.

#### Parameters

out	<i>max_cert_size</i>	Maximum certificate size will be returned here in bytes.
-----	----------------------	--

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**20.200.2.3 tng\_atcacert\_read\_device\_cert()**

```
int tng_atcacert_read_device_cert (
    uint8_t * cert,
    size_t * cert_size,
    const uint8_t * signer_cert )
```

Reads the device certificate for a TNG device.

**Parameters**

out	<i>cert</i>	Buffer to received the certificate (DER format).
in, out	<i>cert_size</i>	As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in cert in bytes.
in	<i>signer_cert</i>	If supplied, the signer public key is used from this certificate. If set to NULL, the signer public key is read from the device.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**20.200.2.4 tng\_atcacert\_read\_signer\_cert()**

```
int tng_atcacert_read_signer_cert (
    uint8_t * cert,
    size_t * cert_size )
```

Reads the signer certificate for a TNG device.

**Parameters**

out	<i>cert</i>	Buffer to received the certificate (DER format).
in, out	<i>cert_size</i>	As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in cert in bytes.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 20.200.2.5 tng\_atcacert\_root\_cert()

```
int tng_atcacert_root_cert (
    uint8_t * cert,
    size_t * cert_size )
```

Get the TNG root cert.

#### Parameters

out	<i>cert</i>	Buffer to received the certificate (DER format).
in, out	<i>cert_size</i>	As input, the size of the cert buffer in bytes. As output, the size of the certificate returned in cert in bytes.

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 20.200.2.6 tng\_atcacert\_root\_cert\_size()

```
int tng_atcacert_root_cert_size (
    size_t * cert_size )
```

Get the size of the TNG root cert.

#### Parameters

out	<i>cert_size</i>	Certificate size will be returned here in bytes.
-----	------------------	--

#### Returns

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

### 20.200.2.7 tng\_atcacert\_root\_public\_key()

```
int tng_atcacert_root_public_key (
    uint8_t * public_key )
```

Gets the root public key.

#### Parameters

out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve.
-----	-------------------	--

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**20.200.2.8 tng\_atcacert\_signer\_public\_key()**

```
int tng_atcacert_signer_public_key (
    uint8_t * public_key,
    uint8_t * cert )
```

Reads the signer public key.

**Parameters**

out	<i>public_key</i>	Public key will be returned here. Format will be the X and Y integers in big-endian format. 64 bytes for P256 curve.
in	<i>cert</i>	If supplied, the signer public key is used from this certificate. If set to NULL, the signer public key is read from the device.

**Returns**

ATCACERT\_E\_SUCCESS on success, otherwise an error code.

**20.201 tng\_atcacert\_client.h File Reference**

Client side certificate I/O functions for TNG devices.

```
#include <stdint.h>
#include "atcacert/atcacert.h"
```

**Functions**

- int [tng\\_atcacert\\_max\\_device\\_cert\\_size](#) (size\_t \*max\_cert\_size)  
*Return the maximum possible certificate size in bytes for a TNG device certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.*
- int [tng\\_atcacert\\_read\\_device\\_cert](#) (uint8\_t \*cert, size\_t \*cert\_size, const uint8\_t \*signer\_cert)  
*Reads the device certificate for a TNG device.*
- int [tng\\_atcacert\\_device\\_public\\_key](#) (uint8\_t \*public\_key, uint8\_t \*cert)  
*Reads the device public key.*
- int [tng\\_atcacert\\_max\\_signer\\_cert\\_size](#) (size\_t \*max\_cert\_size)  
*Return the maximum possible certificate size in bytes for a TNG signer certificate. Certificate can be variable size, so this gives an appropriate buffer size when reading the certificate.*
- int [tng\\_atcacert\\_read\\_signer\\_cert](#) (uint8\_t \*cert, size\_t \*cert\_size)

## 20.202 tng\_root\_cert.c File Reference

---

- Reads the signer certificate for a TNG device.*
  - int [tng\\_atcacert\\_signer\\_public\\_key](#) (uint8\_t \*public\_key, uint8\_t \*cert)  
*Reads the signer public key.*
- int [tng\\_atcacert\\_root\\_cert\\_size](#) (size\_t \*cert\_size)  
*Get the size of the TNG root cert.*
- int [tng\\_atcacert\\_root\\_cert](#) (uint8\_t \*cert, size\_t \*cert\_size)  
*Get the TNG root cert.*
- int [tng\\_atcacert\\_root\\_public\\_key](#) (uint8\_t \*public\_key)  
*Gets the root public key.*

### 20.201.1 Detailed Description

Client side certificate I/O functions for TNG devices.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.202 tng\_root\_cert.c File Reference

TNG root certificate (DER)

```
#include <stdint.h>
#include <stddef.h>
```

### Variables

- const uint8\_t [g\\_cryptoauth\\_root\\_ca\\_002\\_cert](#) [501]
- const size\_t [g\\_cryptoauth\\_root\\_ca\\_002\\_cert\\_size](#) = sizeof([g\\_cryptoauth\\_root\\_ca\\_002\\_cert](#))

### 20.202.1 Detailed Description

TNG root certificate (DER)

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.202.2 Variable Documentation



### 20.202.2.1 g\_cryptoauth\_root\_ca\_002\_cert

```
const uint8_t g_cryptoauth_root_ca_002_cert[501]
```

### 20.202.2.2 g\_cryptoauth\_root\_ca\_002\_cert\_size

```
const size_t g_cryptoauth_root_ca_002_cert_size = sizeof(g_cryptoauth_root_ca_002_cert)
```

## 20.203 tng\_root\_cert.h File Reference

TNG root certificate (DER)

```
#include <stdint.h>
```

- #define [CRYPTOAUTH\\_ROOT\\_CA\\_002\\_PUBLIC\\_KEY\\_OFFSET](#) 266
- const uint8\_t [g\\_cryptoauth\\_root\\_ca\\_002\\_cert](#) []
- const size\_t [g\\_cryptoauth\\_root\\_ca\\_002\\_cert\\_size](#)

### 20.203.1 Detailed Description

TNG root certificate (DER)

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.204 tnglora\_cert\_def\_1\_signer.c File Reference

TNG LORA signer certificate definition.

```
#include "atcacert/atcacert_def.h"  
#include "tngtls_cert_def_1_signer.h"
```

### Variables

- const uint8\_t [g\\_tngtls\\_cert\\_template\\_1\\_signer](#) []
- const [atcacert\\_cert\\_element\\_t](#) [g\\_tngtls\\_cert\\_elements\\_1\\_signer](#) []
- const [atcacert\\_def\\_t](#) [g\\_tnglora\\_cert\\_def\\_1\\_signer](#)

### 20.204.1 Detailed Description

TNG LORA signer certificate definition.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.204.2 Variable Documentation

#### 20.204.2.1 g\_tngtls\_cert\_elements\_1\_signer

```
const atccert_cert_element_t g_tngtls_cert_elements_1_signer[]
```

#### 20.204.2.2 g\_tngtls\_cert\_template\_1\_signer

```
const uint8_t g_tngtls_cert_template_1_signer[]
```

## 20.205 tnglora\_cert\_def\_1\_signer.h File Reference

TNG LORA signer certificate definition.

```
#include "atccert/atccert_def.h"
```

### Variables

- const atccert\_def\_t g\_tnglora\_cert\_def\_1\_signer

### 20.205.1 Detailed Description

TNG LORA signer certificate definition.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.206 tnglora\_cert\_def\_2\_device.c File Reference

TNG LORA device certificate definition.

```
#include "atcacert/atcacert_def.h"
#include "tngtls_cert_def_2_device.h"
#include "tnglora_cert_def_1_signer.h"
```

### Variables

- const uint8\_t [g\\_tngtls\\_cert\\_template\\_2\\_device](#) []
- const [atcacert\\_cert\\_element\\_t](#) [g\\_tngtls\\_cert\\_elements\\_2\\_device](#) []
- const [atcacert\\_def\\_t](#) [g\\_tnglora\\_cert\\_def\\_2\\_device](#)

### 20.206.1 Detailed Description

TNG LORA device certificate definition.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.206.2 Variable Documentation

#### 20.206.2.1 g\_tngtls\_cert\_elements\_2\_device

```
const atcacert\_cert\_element\_t g_tngtls_cert_elements_2_device[]
```

#### 20.206.2.2 g\_tngtls\_cert\_template\_2\_device

```
const uint8_t g_tngtls_cert_template_2_device[]
```

## 20.207 tnglora\_cert\_def\_2\_device.h File Reference

TNG LORA device certificate definition.

```
#include "atcacert/atcacert_def.h"
```

### Variables

- const [atcacert\\_def\\_t g\\_tnglora\\_cert\\_def\\_2\\_device](#)

### 20.207.1 Detailed Description

TNG LORA device certificate definition.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.208 tnglora\_cert\_def\_4\_device.c File Reference

TNG LORA device certificate definition.

```
#include "atcacert/atcacert_def.h"
#include "tnglora_cert_def_4_device.h"
#include "tnglora_cert_def_1_signer.h"
```

### Variables

- const uint8\_t [g\\_tnglora\\_cert\\_template\\_4\\_device](#) [552]
- const [atcacert\\_cert\\_element\\_t g\\_tnglora\\_cert\\_elements\\_4\\_device](#) []
- const [atcacert\\_def\\_t g\\_tnglora\\_cert\\_def\\_4\\_device](#)

### 20.208.1 Detailed Description

TNG LORA device certificate definition.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.208.2 Variable Documentation

#### 20.208.2.1 g\_tnglora\_cert\_def\_4\_device

```
const atcacert\_def\_t g_tnglora_cert_def_4_device
```

### 20.208.2.2 g\_tnqlora\_cert\_elements\_4\_device

```
const atcacert_cert_element_t g_tnqlora_cert_elements_4_device[]
```

### 20.208.2.3 g\_tnqlora\_cert\_template\_4\_device

```
const uint8_t g_tnqlora_cert_template_4_device[552]
```

## 20.209 tnqlora\_cert\_def\_4\_device.h File Reference

TNG LORA device certificate definition.

```
#include "atcacert/atcacert_def.h"
```

- #define [TNQLORA\\_CERT\\_TEMPLATE\\_4\\_DEVICE\\_SIZE](#) 552
- const [atcacert\\_def\\_t g\\_tnqlora\\_cert\\_def\\_4\\_device](#)

### 20.209.1 Detailed Description

TNG LORA device certificate definition.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.210 tngtls\_cert\_def\_1\_signer.c File Reference

TNG TLS signer certificate definition.

```
#include "atcacert/atcacert_def.h"
#include "tngtls_cert_def_1_signer.h"
```

### Variables

- const uint8\_t [g\\_tngtls\\_cert\\_template\\_1\\_signer](#) [520]
- const [atcacert\\_cert\\_element\\_t g\\_tngtls\\_cert\\_elements\\_1\\_signer](#) []
- const [atcacert\\_def\\_t g\\_tngtls\\_cert\\_def\\_1\\_signer](#)

### 20.210.1 Detailed Description

TNG TLS signer certificate definition.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

### 20.210.2 Variable Documentation

#### 20.210.2.1 g\_tngtls\_cert\_def\_1\_signer

```
const atcacert_def_t g_tngtls_cert_def_1_signer
```

#### 20.210.2.2 g\_tngtls\_cert\_elements\_1\_signer

```
const atcacert_cert_element_t g_tngtls_cert_elements_1_signer[]
```

##### Initial value:

```
= {  
    {  
        .id = "subject",  
        .device_loc = {  
            .zone = DEVZONE_NONE,  
        },  
        .cert_loc = {  
            .offset = 158,  
            .count = 81  
        }  
    }  
}
```

#### 20.210.2.3 g\_tngtls\_cert\_template\_1\_signer

```
const uint8_t g_tngtls_cert_template_1_signer[520]
```

## 20.211 tngtls\_cert\_def\_1\_signer.h File Reference

TNG TLS signer certificate definition.

```
#include "atcacert/atcacert_def.h"
```

- `#define TNGTLS_CERT_TEMPLATE_1_SIGNER_SIZE 520`
- `ATCA_DLL const atcacert_def_t g_tngtls_cert_def_1_signer`

### 20.211.1 Detailed Description

TNG TLS signer certificate definition.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.212 tngtls\_cert\_def\_2\_device.c File Reference

TNG TLS device certificate definition.

```
#include "atcacert/atcacert_def.h"
#include "tngtls_cert_def_2_device.h"
#include "tngtls_cert_def_1_signer.h"
```

### Variables

- const uint8\_t [g\\_tngtls\\_cert\\_template\\_2\\_device](#) [505]
- const [atcacert\\_cert\\_element\\_t g\\_tngtls\\_cert\\_elements\\_2\\_device](#) [2]
- const [atcacert\\_def\\_t g\\_tngtls\\_cert\\_def\\_2\\_device](#)

### 20.212.1 Detailed Description

TNG TLS device certificate definition.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.212.2 Variable Documentation

### 20.212.2.1 g\_tngtls\_cert\_def\_2\_device

```
const atcacert\_def\_t g_tngtls_cert_def_2_device
```

### 20.212.2.2 g\_tngtls\_cert\_elements\_2\_device

```
const atcacert\_cert\_element\_t g_tngtls_cert_elements_2_device[2]
```

## 20.213 tngtls\_cert\_def\_2\_device.h File Reference

---

### 20.212.2.3 g\_tngtls\_cert\_template\_2\_device

```
const uint8_t g_tngtls_cert_template_2_device[505]
```

## 20.213 tngtls\_cert\_def\_2\_device.h File Reference

TNG TLS device certificate definition.

```
#include "atcacert/atcacert_def.h"
```

- `#define TNGTLS_CERT_TEMPLATE_2_DEVICE_SIZE` 505
- `#define TNGTLS_CERT_ELEMENTS_2_DEVICE_COUNT` 2
- `const atcacert_def_t g_tngtls_cert_def_2_device`

### 20.213.1 Detailed Description

TNG TLS device certificate definition.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.214 tngtls\_cert\_def\_3\_device.c File Reference

TNG TLS device certificate definition.

```
#include "atcacert/atcacert_def.h"  
#include "tngtls_cert_def_3_device.h"  
#include "tngtls_cert_def_1_signer.h"
```

### Variables

- `const uint8_t g_tngtls_cert_template_3_device` [546]
- `const atcacert_cert_element_t g_tngtls_cert_elements_3_device` []
- `const atcacert_def_t g_tngtls_cert_def_3_device`

### 20.214.1 Detailed Description

TNG TLS device certificate definition.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.



## 20.214.2 Variable Documentation

### 20.214.2.1 g\_tngtls\_cert\_def\_3\_device

```
const atcacert_def_t g_tngtls_cert_def_3_device
```

### 20.214.2.2 g\_tngtls\_cert\_elements\_3\_device

```
const atcacert_cert_element_t g_tngtls_cert_elements_3_device[ ]
```

### 20.214.2.3 g\_tngtls\_cert\_template\_3\_device

```
const uint8_t g_tngtls_cert_template_3_device[546]
```

## 20.215 tngtls\_cert\_def\_3\_device.h File Reference

TNG TLS device certificate definition.

```
#include "atcacert/atcacert_def.h"
```

- #define [TNGTLS\\_CERT\\_TEMPLATE\\_3\\_DEVICE\\_SIZE](#) 546
- const [atcacert\\_def\\_t g\\_tngtls\\_cert\\_def\\_3\\_device](#)

### 20.215.1 Detailed Description

TNG TLS device certificate definition.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

## 20.216 trust\_pkcs11\_config.c File Reference

PKCS11 Trust Platform Configuration.

```
#include "cryptoauthlib.h"
#include "pkcs11_config.h"
#include "pkcs11/pkcs11_object.h"
#include "pkcs11/pkcs11_slot.h"
```

### 20.216.1 Detailed Description

PKCS11 Trust Platform Configuration.

#### Copyright

(c) 2015-2020 Microchip Technology Inc. and its subsidiaries.

# Index

- `_NOP`
  - `sha1_routines.h`, [1072](#)
- `_WDRESET`
  - `sha1_routines.h`, [1072](#)
- `__PASTE`
  - `pkcs11.h`, [894](#)
- `_atcab_exit`
  - Basic Crypto API methods (`atcab_`), [60](#)
- `_atecc508a_config`, [387](#)
  - `ChipMode`, [387](#)
  - `Counter0`, [388](#)
  - `Counter1`, [388](#)
  - `I2C_Address`, [388](#)
  - `I2C_Enable`, [388](#)
  - `KeyConfig`, [388](#)
  - `LastKeyUse`, [388](#)
  - `LockConfig`, [388](#)
  - `LockValue`, [388](#)
  - `OTPmode`, [389](#)
  - `Reserved0`, [389](#)
  - `Reserved1`, [389](#)
  - `Reserved2`, [389](#)
  - `RevNum`, [389](#)
  - `RFU`, [389](#)
  - `Selector`, [389](#)
  - `SlotConfig`, [389](#)
  - `SlotLocked`, [390](#)
  - `SN03`, [390](#)
  - `SN47`, [390](#)
  - `SN8`, [390](#)
  - `UserExtra`, [390](#)
  - `X509format`, [390](#)
- `_atecc608_config`, [390](#)
  - `AES_Enable`, [391](#)
  - `ChipMode`, [391](#)
  - `ChipOptions`, [391](#)
  - `Counter0`, [392](#)
  - `Counter1`, [392](#)
  - `CountMatch`, [392](#)
  - `I2C_Address`, [392](#)
  - `I2C_Enable`, [392](#)
  - `KdfIvLoc`, [392](#)
  - `KdfIvStr`, [392](#)
  - `KeyConfig`, [392](#)
  - `LockConfig`, [393](#)
  - `LockValue`, [393](#)
  - `Reserved1`, [393](#)
  - `Reserved2`, [393](#)
  - `Reserved3`, [393](#)
  - `RevNum`, [393](#)
  - `SecureBoot`, [393](#)
  - `SlotConfig`, [393](#)
  - `SlotLocked`, [394](#)
  - `SN03`, [394](#)
  - `SN47`, [394](#)
  - `SN8`, [394](#)
  - `UseLock`, [394](#)
  - `UserExtra`, [394](#)
  - `UserExtraAdd`, [394](#)
  - `VolatileKeyPermission`, [394](#)
  - `X509format`, [395](#)
- `_atsha204a_config`, [395](#)
  - `ChipMode`, [395](#)
  - `Counter`, [395](#)
  - `I2C_Address`, [396](#)
  - `I2C_Enable`, [396](#)
  - `LastKeyUse`, [396](#)
  - `LockConfig`, [396](#)
  - `LockValue`, [396](#)
  - `OTPmode`, [396](#)
  - `Reserved0`, [396](#)
  - `Reserved1`, [396](#)
  - `Reserved2`, [397](#)
  - `RevNum`, [397](#)
  - `Selector`, [397](#)
  - `SlotConfig`, [397](#)
  - `SN03`, [397](#)
  - `SN47`, [397](#)
  - `SN8`, [397](#)
  - `UserExtra`, [397](#)
- `_calib_exit`
  - Basic Crypto API methods for CryptoAuth Devices (`calib_`), [206](#)
- `_gDevice`
  - `atca_basic.c`, [563](#)
  - Basic Crypto API methods (`atcab_`), [120](#)
- `_gHid`
  - Hardware abstraction layer (`hal_`), [307](#)
- `_pkcs11_mech_table_e`, [398](#)
  - `info`, [398](#)
  - `type`, [398](#)
- `_pkcs11_attr_model`, [398](#)
  - `func`, [398](#)
  - `type`, [398](#)
- `_pkcs11_lib_ctx`, [399](#)
  - `config_path`, [399](#)
  - `create_mutex`, [399](#)
  - `destroy_mutex`, [399](#)

- initialized, 399
- lock\_mutex, 400
- mutex, 400
- slot\_cnt, 400
- slots, 400
- unlock\_mutex, 400
- \_pkcs11\_object, 400
  - attributes, 401
  - class\_id, 401
  - class\_type, 401
  - config, 401
  - count, 401
  - data, 401
  - flags, 402
  - handle\_info, 402
  - name, 402
  - size, 402
  - slot, 402
- \_pkcs11\_object\_cache\_t, 402
  - handle, 403
  - object, 403
- \_pkcs11\_session\_ctx, 403
  - active\_object, 403
  - attrib\_count, 404
  - attrib\_list, 404
  - error, 404
  - handle, 404
  - initialized, 404
  - logged\_in, 404
  - object\_count, 404
  - object\_index, 404
  - read\_key, 405
  - slot, 405
  - state, 405
- \_pkcs11\_slot\_ctx, 405
  - cfg\_zone, 406
  - device\_ctx, 406
  - flags, 406
  - initialized, 406
  - interface\_config, 406
  - label, 406
  - session, 406
  - slot\_id, 406
  - so\_pin\_handle, 407
  - user\_pin\_handle, 407
- \_reserved
  - ATCAPacket, 464
- aad\_size
  - atca\_aes\_gcm\_ctx, 411
- ACK\_CHECK\_DIS
  - hal\_esp32\_i2c.c, 836
- ACK\_CHECK\_EN
  - hal\_esp32\_i2c.c, 836
- ACK\_VAL
  - hal\_esp32\_i2c.c, 836
- active\_object
  - \_pkcs11\_session\_ctx, 403
- AES\_COUNT
  - calib\_command.h, 717
- AES\_DATA\_SIZE
  - calib\_command.h, 717
- AES\_Enable
  - \_atecc608\_config, 391
- AES\_INPUT\_IDX
  - calib\_command.h, 717
- AES\_KEYID\_IDX
  - calib\_command.h, 717
- AES\_MODE\_DECRYPT
  - calib\_command.h, 717
- AES\_MODE\_ENCRYPT
  - calib\_command.h, 717
- AES\_MODE\_GFM
  - calib\_command.h, 718
- AES\_MODE\_IDX
  - calib\_command.h, 718
- AES\_MODE\_KEY\_BLOCK\_MASK
  - calib\_command.h, 718
- AES\_MODE\_KEY\_BLOCK\_POS
  - calib\_command.h, 718
- AES\_MODE\_MASK
  - calib\_command.h, 718
- AES\_MODE\_OP\_MASK
  - calib\_command.h, 718
- AES\_RSP\_SIZE
  - calib\_command.h, 719
- ANY
  - license.txt, 884
- api\_206a.c, 543
  - sha206a\_authenticate, 544
  - sha206a\_check\_dk\_useflag\_validity, 544
  - sha206a\_check\_pk\_useflag\_validity, 545
  - sha206a\_diversify\_parent\_key, 545
  - sha206a\_generate\_challenge\_response\_pair, 545
  - sha206a\_generate\_derive\_key, 546
  - sha206a\_get\_data\_store\_lock\_status, 546
  - sha206a\_get\_dk\_update\_count, 547
  - sha206a\_get\_dk\_useflag\_count, 547
  - sha206a\_get\_pk\_useflag\_count, 547
  - sha206a\_read\_data\_store, 548
  - sha206a\_verify\_device\_consumption, 548
  - sha206a\_write\_data\_store, 549
- api\_206a.h, 549
  - ATCA\_SHA206A\_DKEY\_CONSUMPTION\_MASK, 550
  - ATCA\_SHA206A\_PKEY\_CONSUMPTION\_MASK, 550
  - ATCA\_SHA206A\_SYMMETRIC\_KEY\_ID\_SLOT, 551
  - ATCA\_SHA206A\_ZONE\_WRITE\_LOCK, 551
  - sha206a\_authenticate, 551
  - sha206a\_check\_dk\_useflag\_validity, 552
  - sha206a\_check\_pk\_useflag\_validity, 552
  - SHA206A\_DATA\_STORE0, 551
  - SHA206A\_DATA\_STORE1, 551
  - SHA206A\_DATA\_STORE2, 551
  - sha206a\_diversify\_parent\_key, 552

- sha206a\_generate\_challenge\_response\_pair, 553
- sha206a\_generate\_derive\_key, 553
- sha206a\_get\_data\_store\_lock\_status, 554
- sha206a\_get\_dk\_update\_count, 554
- sha206a\_get\_dk\_useflag\_count, 554
- sha206a\_get\_pk\_useflag\_count, 555
- sha206a\_read\_data\_store, 555
- sha206a\_verify\_device\_consumption, 556
- sha206a\_write\_data\_store, 556
- app\_digest
  - secure\_boot\_parameters, 540
- atAES
  - calib\_command.c, 687
  - calib\_command.h, 788
- ATCA\_ADDRESS\_MASK
  - calib\_command.h, 719
- ATCA\_ADDRESS\_MASK\_CONFIG
  - calib\_command.h, 719
- ATCA\_ADDRESS\_MASK\_OTP
  - calib\_command.h, 719
- ATCA\_AES
  - calib\_command.h, 719
- ATCA\_AES128\_BLOCK\_SIZE
  - cryptoauthlib.h, 823
- ATCA\_AES128\_KEY\_SIZE
  - cryptoauthlib.h, 823
- atca\_aes\_cbc\_ctx, 407
  - ciphertext, 407
  - device, 407
  - key\_block, 408
  - key\_id, 408
- atca\_aes\_cbc\_ctx\_t
  - atca\_crypto\_hw\_aes.h, 576
- atca\_aes\_cmac\_ctx, 408
  - block, 408
  - block\_size, 408
  - cbc\_ctx, 409
- atca\_aes\_cmac\_ctx\_t
  - atca\_crypto\_hw\_aes.h, 576
- atca\_aes\_ctr\_ctx, 409
  - cb, 409
  - counter\_size, 409
  - device, 410
  - key\_block, 410
  - key\_id, 410
- atca\_aes\_ctr\_ctx\_t
  - atca\_crypto\_hw\_aes.h, 576
- ATCA\_AES\_ENABLE\_EN\_MASK
  - ATCADevice (atca\_), 127
- ATCA\_AES\_ENABLE\_EN\_SHIFT
  - ATCADevice (atca\_), 127
- atca\_aes\_gcm\_ctx, 410
  - aad\_size, 411
  - cb, 411
  - ciphertext\_block, 411
  - data\_size, 412
  - enc\_cb, 412
  - h, 412
  - j0, 412
  - key\_block, 412
  - key\_id, 412
  - partial\_aad, 413
  - partial\_aad\_size, 413
  - y, 413
- atca\_aes\_gcm\_ctx\_t
  - Basic Crypto API methods (atcab\_), 59
- ATCA\_AES\_GCM\_IV\_STD\_LENGTH
  - Basic Crypto API methods (atcab\_), 59
- ATCA\_AES\_GFM\_SIZE
  - calib\_command.h, 719
- ATCA\_AES\_KEY\_TYPE
  - calib\_command.h, 720
- ATCA\_ALLOC\_FAILURE
  - atca\_status.h, 646
- ATCA\_ASSERT\_FAILURE
  - atca\_status.h, 645
- ATCA\_ATECC608\_SUPPORT
  - cryptoauthlib.h, 824
- ATCA\_B283\_KEY\_TYPE
  - calib\_command.h, 720
- ATCA\_BAD\_OPCODE
  - atca\_status.h, 645
- ATCA\_BAD\_PARAM
  - atca\_status.h, 645
- atca\_basic.c, 557
  - \_gDevice, 563
  - atca\_version, 563
- atca\_basic.h, 563
- atca\_basic\_aes\_gcm\_version
  - Basic Crypto API methods (atcab\_), 120
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 250
- ATCA\_BLOCK\_SIZE
  - calib\_command.h, 720
- atca\_bool.h, 571
- ATCA\_CA\_SUPPORT
  - cryptoauthlib.h, 824
- atca\_cfgs.c, 571
- atca\_cfgs.h, 571
  - cfg\_ateccx08a\_i2c\_default, 572
  - cfg\_ateccx08a\_kitcdc\_default, 572
  - cfg\_ateccx08a\_kithid\_default, 572
  - cfg\_ateccx08a\_swi\_default, 573
  - cfg\_atsha20xa\_i2c\_default, 573
  - cfg\_atsha20xa\_kitcdc\_default, 573
  - cfg\_atsha20xa\_kithid\_default, 573
  - cfg\_atsha20xa\_swi\_default, 573
- atca\_check\_mac\_in\_out, 413
  - client\_chal, 414
  - client\_resp, 414
  - key\_id, 414
  - mode, 414
  - other\_data, 414
  - otp, 414
  - slot\_key, 415
  - sn, 415

target\_key, 415  
 temp\_key, 415  
 atca\_check\_mac\_in\_out\_t  
   Host side crypto methods (atcah\_), 315  
 ATCA\_CHECKMAC  
   calib\_command.h, 720  
 ATCA\_CHECKMAC\_VERIFY\_FAILED  
   atca\_status.h, 645  
 ATCA\_CHIP\_MODE\_CLK\_DIV  
   ATCADevice (atca\_), 127  
 ATCA\_CHIP\_MODE\_CLK\_DIV\_MASK  
   ATCADevice (atca\_), 127  
 ATCA\_CHIP\_MODE\_CLK\_DIV\_SHIFT  
   ATCADevice (atca\_), 127  
 ATCA\_CHIP\_MODE\_I2C\_EXTRA\_MASK  
   ATCADevice (atca\_), 127  
 ATCA\_CHIP\_MODE\_I2C\_EXTRA\_SHIFT  
   ATCADevice (atca\_), 127  
 ATCA\_CHIP\_MODE\_TTL\_EN\_MASK  
   ATCADevice (atca\_), 128  
 ATCA\_CHIP\_MODE\_TTL\_EN\_SHIFT  
   ATCADevice (atca\_), 128  
 ATCA\_CHIP\_MODE\_WDG\_LONG\_MASK  
   ATCADevice (atca\_), 128  
 ATCA\_CHIP\_MODE\_WDG\_LONG\_SHIFT  
   ATCADevice (atca\_), 128  
 ATCA\_CHIP\_OPT\_ECDH\_PROT  
   ATCADevice (atca\_), 128  
 ATCA\_CHIP\_OPT\_ECDH\_PROT\_MASK  
   ATCADevice (atca\_), 128  
 ATCA\_CHIP\_OPT\_ECDH\_PROT\_SHIFT  
   ATCADevice (atca\_), 128  
 ATCA\_CHIP\_OPT\_IO\_PROT\_EN\_MASK  
   ATCADevice (atca\_), 129  
 ATCA\_CHIP\_OPT\_IO\_PROT\_EN\_SHIFT  
   ATCADevice (atca\_), 129  
 ATCA\_CHIP\_OPT\_IO\_PROT\_KEY  
   ATCADevice (atca\_), 129  
 ATCA\_CHIP\_OPT\_IO\_PROT\_KEY\_MASK  
   ATCADevice (atca\_), 129  
 ATCA\_CHIP\_OPT\_IO\_PROT\_KEY\_SHIFT  
   ATCADevice (atca\_), 129  
 ATCA\_CHIP\_OPT\_KDF\_AES\_EN\_MASK  
   ATCADevice (atca\_), 129  
 ATCA\_CHIP\_OPT\_KDF\_AES\_EN\_SHIFT  
   ATCADevice (atca\_), 129  
 ATCA\_CHIP\_OPT\_KDF\_PROT  
   ATCADevice (atca\_), 130  
 ATCA\_CHIP\_OPT\_KDF\_PROT\_MASK  
   ATCADevice (atca\_), 130  
 ATCA\_CHIP\_OPT\_KDF\_PROT\_SHIFT  
   ATCADevice (atca\_), 130  
 ATCA\_CHIP\_OPT\_POST\_EN\_MASK  
   ATCADevice (atca\_), 130  
 ATCA\_CHIP\_OPT\_POST\_EN\_SHIFT  
   ATCADevice (atca\_), 130  
 ATCA\_CHIPMODE\_CLOCK\_DIV\_M0  
   calib\_command.h, 720  
 ATCA\_CHIPMODE\_CLOCK\_DIV\_M1  
   calib\_command.h, 720  
 ATCA\_CHIPMODE\_CLOCK\_DIV\_M2  
   calib\_command.h, 721  
 ATCA\_CHIPMODE\_CLOCK\_DIV\_MASK  
   calib\_command.h, 721  
 ATCA\_CHIPMODE\_I2C\_ADDRESS\_FLAG  
   calib\_command.h, 721  
 ATCA\_CHIPMODE\_OFFSET  
   calib\_command.h, 721  
 ATCA\_CHIPMODE\_TTL\_ENABLE\_FLAG  
   calib\_command.h, 721  
 ATCA\_CHIPMODE\_WATCHDOG\_LONG  
   calib\_command.h, 721  
 ATCA\_CHIPMODE\_WATCHDOG\_MASK  
   calib\_command.h, 722  
 ATCA\_CHIPMODE\_WATCHDOG\_SHORT  
   calib\_command.h, 722  
 ATCA\_CMD\_SIZE\_MAX  
   calib\_command.h, 722  
 ATCA\_CMD\_SIZE\_MIN  
   calib\_command.h, 722  
 ATCA\_COMM\_FAIL  
   atca\_status.h, 645  
 atca\_command, 415  
   clock\_divider, 416  
   dt, 416  
   execution\_time\_msec, 416  
 atca\_command.c, 574  
 atca\_command.h, 574  
 ATCA\_COMMAND\_HEADER\_SIZE  
   Host side crypto methods (atcah\_), 312  
 atca\_compiler.h, 575  
 ATCA\_CONFIG\_ZONE\_LOCKED  
   atca\_status.h, 645  
 ATCA\_COUNT\_IDX  
   calib\_command.h, 722  
 ATCA\_COUNT\_SIZE  
   calib\_command.h, 722  
 ATCA\_COUNTER  
   calib\_command.h, 723  
 ATCA\_COUNTER\_MATCH\_EN\_MASK  
   ATCADevice (atca\_), 130  
 ATCA\_COUNTER\_MATCH\_EN\_SHIFT  
   ATCADevice (atca\_), 130  
 ATCA\_COUNTER\_MATCH\_KEY  
   ATCADevice (atca\_), 131  
 ATCA\_COUNTER\_MATCH\_KEY\_MASK  
   ATCADevice (atca\_), 131  
 ATCA\_COUNTER\_MATCH\_KEY\_SHIFT  
   ATCADevice (atca\_), 131  
 ATCA\_CRC\_SIZE  
   calib\_command.h, 723  
 atca\_crypto\_hw\_aes.h, 575  
   atca\_aes\_cbc\_ctx\_t, 576  
   atca\_aes\_cmac\_ctx\_t, 576  
   atca\_aes\_ctr\_ctx\_t, 576  
 atca\_crypto\_hw\_aes\_cbc.c, 576

[atca\\_crypto\\_hw\\_aes\\_cmac.c](#), 577  
[atca\\_crypto\\_hw\\_aes\\_ctr.c](#), 578  
[atca\\_crypto\\_sw.h](#), 579  
     ATCA\_SHA1\_DIGEST\_SIZE, 580  
     ATCA\_SHA2\_256\_BLOCK\_SIZE, 580  
     ATCA\_SHA2\_256\_DIGEST\_SIZE, 581  
     atcac\_aes\_cmac\_ctx, 581  
     atcac\_aes\_cmac\_finish, 582  
     atcac\_aes\_cmac\_init, 582  
     atcac\_aes\_cmac\_update, 582  
     atcac\_aes\_gcm\_aad\_update, 582  
     atcac\_aes\_gcm\_ctx, 581  
     atcac\_aes\_gcm\_decrypt\_finish, 583  
     atcac\_aes\_gcm\_decrypt\_start, 583  
     atcac\_aes\_gcm\_decrypt\_update, 583  
     atcac\_aes\_gcm\_encrypt\_finish, 584  
     atcac\_aes\_gcm\_encrypt\_start, 584  
     atcac\_aes\_gcm\_encrypt\_update, 584  
     atcac\_hmac\_sha256\_ctx, 581  
     atcac\_sha1\_ctx, 581  
     atcac\_sha2\_256\_ctx, 581  
     MBEDTLS\_CMAC\_C, 581  
[atca\\_crypto\\_sw\\_ecdsa.c](#), 585  
[atca\\_crypto\\_sw\\_ecdsa.h](#), 585  
[atca\\_crypto\\_sw\\_rand.c](#), 586  
[atca\\_crypto\\_sw\\_rand.h](#), 586  
[atca\\_crypto\\_sw\\_sha1.c](#), 587  
[atca\\_crypto\\_sw\\_sha1.h](#), 587  
[atca\\_crypto\\_sw\\_sha2.c](#), 588  
[atca\\_crypto\\_sw\\_sha2.h](#), 588  
 ATCA\_CUSTOM\_IFACE  
     ATCAIface (atca\_), 147  
 ATCA\_DATA\_IDX  
     calib\_command.h, 723  
 ATCA\_DATA\_SIZE  
     calib\_command.h, 723  
 ATCA\_DATA\_ZONE\_LOCKED  
     atca\_status.h, 645  
[atca\\_debug.c](#), 589  
     atca\_trace, 590  
     atca\_trace\_config, 590  
     atca\_trace\_msg, 590  
     g\_trace\_fp, 590  
[atca\\_debug.h](#), 590  
     atca\_trace, 591  
     atca\_trace\_config, 591  
     atca\_trace\_msg, 591  
[atca\\_decrypt\\_in\\_out](#), 416  
[atca\\_delay\\_10us](#)  
     Hardware abstraction layer (hal\_), 269  
[atca\\_delay\\_ms](#)  
     hal\_esp32\_timer.c, 847  
     Hardware abstraction layer (hal\_), 269  
[atca\\_delay\\_us](#)  
     Hardware abstraction layer (hal\_), 270  
 ATCA\_DERIVE\_KEY  
     calib\_command.h, 723  
[atca\\_derive\\_key\\_in\\_out](#), 417  
     mode, 417  
     parent\_key, 417  
     sn, 417  
     target\_key, 418  
     target\_key\_id, 418  
     temp\_key, 418  
[atca\\_derive\\_key\\_mac\\_in\\_out](#), 418  
     mac, 419  
     mode, 419  
     parent\_key, 419  
     sn, 419  
     target\_key\_id, 419  
 ATCA\_DERIVE\_KEY\_ZEROS\_SIZE  
     Host side crypto methods (atcah\_), 312  
 ATCA\_DEV\_UNKNOWN  
     ATCADevice (atca\_), 143  
[atca\\_device](#), 420  
     mCommands, 420  
     mlface, 420  
     session\_counter, 420  
     session\_key, 420  
     session\_key\_id, 421  
     session\_key\_len, 421  
     session\_state, 421  
[atca\\_device.c](#), 591  
[atca\\_device.h](#), 592  
[atca\\_devtypes.h](#), 595  
 ATCA\_DLL  
     cryptoauthlib.h, 824  
 ATCA\_ECC\_CONFIG\_SIZE  
     calib\_command.h, 723  
 ATCA\_ECC\_P256\_FIELD\_SIZE  
     Software crypto methods (atcac\_), 251  
 ATCA\_ECC\_P256\_PRIVATE\_KEY\_SIZE  
     Software crypto methods (atcac\_), 252  
 ATCA\_ECC\_P256\_PUBLIC\_KEY\_SIZE  
     Software crypto methods (atcac\_), 252  
 ATCA\_ECC\_P256\_SIGNATURE\_SIZE  
     Software crypto methods (atcac\_), 252  
 ATCA\_ECC\_SUPPORT  
     cryptoauthlib.h, 824  
 ATCA\_ECCP256\_KEY\_SIZE  
     cryptoauthlib.h, 824  
 ATCA\_ECCP256\_PUBKEY\_SIZE  
     cryptoauthlib.h, 824  
 ATCA\_ECCP256\_SIG\_SIZE  
     cryptoauthlib.h, 824  
 ATCA\_ECDH  
     calib\_command.h, 724  
[atca\\_execute\\_command](#)  
     Basic Crypto API methods (atcab\_), 59  
 ATCA\_EXECUTION\_ERROR  
     atca\_status.h, 645  
 ATCA\_FUNC\_FAIL  
     atca\_status.h, 645  
[atca\\_gen\\_dig\\_in\\_out](#), 421  
     counter, 422  
     is\_key\_nomac, 422

- key\_conf, 422
- key\_id, 422
- other\_data, 422
- slot\_conf, 422
- slot\_locked, 423
- sn, 423
- stored\_value, 423
- temp\_key, 423
- zone, 423
- atca\_gen\_dig\_in\_out\_t
  - Host side crypto methods (atcah\_), 315
- ATCA\_GEN\_FAIL
  - atca\_status.h, 645
- atca\_gen\_key\_in\_out, 424
  - key\_id, 424
  - mode, 424
  - other\_data, 424
  - public\_key, 425
  - public\_key\_size, 425
  - sn, 425
  - temp\_key, 425
- atca\_gen\_key\_in\_out\_t
  - Host side crypto methods (atcah\_), 315
- ATCA\_GENDIG
  - calib\_command.h, 724
- ATCA\_GENDIG\_ZEROS\_SIZE
  - Host side crypto methods (atcah\_), 312
- ATCA\_GENKEY
  - calib\_command.h, 724
- atca\_hal.c, 596
- atca\_hal.h, 596
- ATCA\_HEALTH\_TEST\_ERROR
  - atca\_status.h, 645
- atca\_helpers.c, 597
  - atcab\_b64rules\_default, 609
  - atcab\_b64rules\_mime, 609
  - atcab\_b64rules\_urlsafes, 609
  - atcab\_base64decode, 599
  - atcab\_base64decode\_, 600
  - atcab\_base64encode, 600
  - atcab\_base64encode\_, 601
  - atcab\_bin2hex, 601
  - atcab\_bin2hex\_, 601
  - atcab\_hex2bin, 602
  - atcab\_hex2bin\_, 602
  - atcab\_memset\_s, 603
  - atcab\_reversal, 603
  - B64\_IS\_EQUAL, 599
  - B64\_IS\_INVALID, 599
  - base64Char, 603
  - base64Index, 605
  - isAlpha, 605
  - isBase64, 605
  - isBase64Digit, 606
  - isDigit, 606
  - isHex, 607
  - isHexAlpha, 607
  - isHexDigit, 607
  - isWhiteSpace, 608
  - packHex, 608
- atca\_helpers.h, 609
  - atcab\_b64rules\_default, 619
  - atcab\_b64rules\_mime, 619
  - atcab\_b64rules\_urlsafes, 619
  - atcab\_base64decode, 610
  - atcab\_base64decode\_, 611
  - atcab\_base64encode, 611
  - atcab\_base64encode\_, 612
  - atcab\_bin2hex, 612
  - atcab\_bin2hex\_, 613
  - atcab\_hex2bin, 613
  - atcab\_hex2bin\_, 614
  - atcab\_memset\_s, 614
  - atcab\_printbin\_label, 614
  - atcab\_printbin\_sp, 614
  - atcab\_reversal, 614
  - base64Char, 615
  - base64Index, 615
  - isAlpha, 616
  - isBase64, 616
  - isBase64Digit, 616
  - isDigit, 617
  - isHex, 617
  - isHexAlpha, 617
  - isHexDigit, 618
  - isWhiteSpace, 618
  - packHex, 619
- ATCA\_HID\_IFACE
  - ATCAIface (atca\_), 147
- ATCA\_HMAC
  - calib\_command.h, 724
- ATCA\_HMAC\_BLOCK\_SIZE
  - Host side crypto methods (atcah\_), 313
- atca\_hmac\_in\_out, 425
- atca\_hmac\_sha256\_ctx\_t
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 206
- atca\_host.c, 620
- atca\_host.h, 621
- ATCA\_I2C\_ENABLE\_EN\_MASK
  - ATCADevice (atca\_), 131
- ATCA\_I2C\_ENABLE\_EN\_SHIFT
  - ATCADevice (atca\_), 131
- ATCA\_I2C\_IFACE
  - ATCAIface (atca\_), 147
- atca\_iface, 426
  - atidle, 426
  - atinit, 427
  - atpostinit, 427
  - atreceive, 427
  - atsend, 427
  - atsleep, 427
  - atwake, 427
  - hal\_data, 427
  - mIfaceCFG, 427
  - mType, 428



- atca\_iface.c, [624](#)
- atca\_iface.h, [625](#)
- atca\_include\_data\_in\_out, [428](#)
  - mode, [428](#)
- ATCA\_INFO
  - calib\_command.h, [724](#)
- ATCA\_INVALID\_ID
  - atca\_status.h, [645](#)
- ATCA\_INVALID\_LENGTH
  - atca\_status.h, [645](#)
- ATCA\_INVALID\_POINTER
  - atca\_status.h, [645](#)
- ATCA\_INVALID\_SIZE
  - atca\_status.h, [645](#)
- atca\_io\_decrypt\_in\_out, [428](#)
  - data, [429](#)
  - data\_size, [429](#)
  - io\_key, [429](#)
  - out\_nonce, [429](#)
- atca\_io\_decrypt\_in\_out\_t
  - Host side crypto methods (atcah\_), [316](#)
- atca\_jwt.c, [627](#)
- atca\_jwt.h, [627](#)
- atca\_jwt\_add\_claim\_numeric
  - JSON Web Token (JWT) methods (atca\_jwt\_), [331](#)
- atca\_jwt\_add\_claim\_string
  - JSON Web Token (JWT) methods (atca\_jwt\_), [331](#)
- atca\_jwt\_check\_payload\_start
  - JSON Web Token (JWT) methods (atca\_jwt\_), [332](#)
- atca\_jwt\_finalize
  - JSON Web Token (JWT) methods (atca\_jwt\_), [332](#)
- atca\_jwt\_init
  - JSON Web Token (JWT) methods (atca\_jwt\_), [332](#)
- atca\_jwt\_t, [430](#)
  - buf, [430](#)
  - buflen, [430](#)
  - cur, [430](#)
- atca\_jwt\_verify
  - JSON Web Token (JWT) methods (atca\_jwt\_), [332](#)
- ATCA\_K283\_KEY\_TYPE
  - calib\_command.h, [724](#)
- ATCA\_KDF
  - calib\_command.h, [725](#)
- ATCA\_KEY\_CONFIG\_AUTH\_KEY
  - ATCADevice (atca\_), [131](#)
- ATCA\_KEY\_CONFIG\_AUTH\_KEY\_MASK
  - ATCADevice (atca\_), [131](#)
- ATCA\_KEY\_CONFIG\_AUTH\_KEY\_SHIFT
  - ATCADevice (atca\_), [132](#)
- ATCA\_KEY\_CONFIG\_KEY\_TYPE
  - ATCADevice (atca\_), [132](#)
- ATCA\_KEY\_CONFIG\_KEY\_TYPE\_MASK
  - ATCADevice (atca\_), [132](#)
- ATCA\_KEY\_CONFIG\_KEY\_TYPE\_SHIFT
  - ATCADevice (atca\_), [132](#)
- ATCA\_KEY\_CONFIG\_LOCKABLE\_MASK
  - ATCADevice (atca\_), [132](#)
- ATCA\_KEY\_CONFIG\_LOCKABLE\_SHIFT
  - ATCADevice (atca\_), [132](#)
- ATCA\_KEY\_CONFIG\_PERSIST\_DISABLE\_MASK
  - ATCADevice (atca\_), [132](#)
- ATCA\_KEY\_CONFIG\_PERSIST\_DISABLE\_SHIFT
  - ATCADevice (atca\_), [133](#)
- ATCA\_KEY\_CONFIG\_PRIVATE\_MASK
  - ATCADevice (atca\_), [133](#)
- ATCA\_KEY\_CONFIG\_PRIVATE\_SHIFT
  - ATCADevice (atca\_), [133](#)
- ATCA\_KEY\_CONFIG\_PUB\_INFO\_MASK
  - ATCADevice (atca\_), [133](#)
- ATCA\_KEY\_CONFIG\_PUB\_INFO\_SHIFT
  - ATCADevice (atca\_), [133](#)
- ATCA\_KEY\_CONFIG\_REQ\_AUTH\_MASK
  - ATCADevice (atca\_), [133](#)
- ATCA\_KEY\_CONFIG\_REQ\_AUTH\_SHIFT
  - ATCADevice (atca\_), [133](#)
- ATCA\_KEY\_CONFIG\_REQ\_RANDOM\_MASK
  - ATCADevice (atca\_), [133](#)
- ATCA\_KEY\_CONFIG\_REQ\_RANDOM\_SHIFT
  - ATCADevice (atca\_), [134](#)
- ATCA\_KEY\_CONFIG\_RFU\_MASK
  - ATCADevice (atca\_), [134](#)
- ATCA\_KEY\_CONFIG\_RFU\_SHIFT
  - ATCADevice (atca\_), [134](#)
- ATCA\_KEY\_CONFIG\_X509\_ID
  - ATCADevice (atca\_), [134](#)
- ATCA\_KEY\_CONFIG\_X509\_ID\_MASK
  - ATCADevice (atca\_), [134](#)
- ATCA\_KEY\_CONFIG\_X509\_ID\_SHIFT
  - ATCADevice (atca\_), [134](#)
- ATCA\_KEY\_COUNT
  - calib\_command.h, [725](#)
- ATCA\_KEY\_ID\_MAX
  - calib\_command.h, [725](#)
- ATCA\_KEY\_SIZE
  - calib\_command.h, [725](#)
- ATCA\_KIT\_AUTO\_IFACE
  - ATCAIface (atca\_), [147](#)
- ATCA\_KIT\_I2C\_IFACE
  - ATCAIface (atca\_), [147](#)
- ATCA\_KIT\_SPI\_IFACE
  - ATCAIface (atca\_), [147](#)
- ATCA\_KIT\_SWI\_IFACE
  - ATCAIface (atca\_), [147](#)
- ATCA\_KIT\_UNKNOWN\_IFACE
  - ATCAIface (atca\_), [147](#)
- ATCA\_LIBRARY\_VERSION\_BUILD
  - atca\_version.h, [646](#)
- ATCA\_LIBRARY\_VERSION\_DATE
  - atca\_version.h, [646](#)
- ATCA\_LIBRARY\_VERSION\_MAJOR
  - atca\_version.h, [646](#)
- ATCA\_LIBRARY\_VERSION\_MINOR
  - atca\_version.h, [647](#)
- ATCA\_LOCK
  - calib\_command.h, [725](#)
- ATCA\_LOCKED



- calib\_command.h, 725
- ATCA\_MAC
  - calib\_command.h, 726
- atca\_mac\_in\_out, 430
- atca\_mac\_in\_out\_t
  - Host side crypto methods (atcah\_), 316
- ATCA\_MAX\_TRANSFORMS
  - atcacert\_def.h, 659
- atca\_mbedtls\_cert\_add
  - atca\_mbedtls\_wrap.c, 630
  - mbedtls Wrapper methods (atca\_mbedtls\_), 333
- atca\_mbedtls\_ecdh.c, 628
- atca\_mbedtls\_ecdh\_ioprot\_cb
  - mbedtls Wrapper methods (atca\_mbedtls\_), 333
- atca\_mbedtls\_ecdh\_slot\_cb
  - mbedtls Wrapper methods (atca\_mbedtls\_), 333
- atca\_mbedtls\_ecdsa.c, 628
- atca\_mbedtls\_pk\_init
  - mbedtls Wrapper methods (atca\_mbedtls\_), 334
- atca\_mbedtls\_wrap.c, 628
  - atca\_mbedtls\_cert\_add, 630
  - atcac\_aes\_cmac\_finish, 631
  - atcac\_aes\_cmac\_init, 631
  - atcac\_aes\_cmac\_update, 632
  - atcac\_aes\_gcm\_aad\_update, 632
  - atcac\_aes\_gcm\_decrypt\_finish, 632
  - atcac\_aes\_gcm\_decrypt\_start, 633
  - atcac\_aes\_gcm\_decrypt\_update, 633
  - atcac\_aes\_gcm\_encrypt\_finish, 634
  - atcac\_aes\_gcm\_encrypt\_start, 634
  - atcac\_aes\_gcm\_encrypt\_update, 635
  - atcac\_sw\_sha1\_finish, 635
  - atcac\_sw\_sha2\_256\_finish, 636
  - mbedtls\_calloc, 630
  - mbedtls\_free, 630
- atca\_mbedtls\_wrap.h, 636
- ATCA\_MSG\_SIZE\_DERIVE\_KEY
  - Host side crypto methods (atcah\_), 313
- ATCA\_MSG\_SIZE\_DERIVE\_KEY\_MAC
  - Host side crypto methods (atcah\_), 313
- ATCA\_MSG\_SIZE\_ENCRYPT\_MAC
  - Host side crypto methods (atcah\_), 313
- ATCA\_MSG\_SIZE\_GEN\_DIG
  - Host side crypto methods (atcah\_), 313
- ATCA\_MSG\_SIZE\_HMAC
  - Host side crypto methods (atcah\_), 313
- ATCA\_MSG\_SIZE\_MAC
  - Host side crypto methods (atcah\_), 313
- ATCA\_MSG\_SIZE\_NONCE
  - Host side crypto methods (atcah\_), 314
- ATCA\_MSG\_SIZE\_PRIVWRITE\_MAC
  - Host side crypto methods (atcah\_), 314
- ATCA\_MUTEX\_TIMEOUT
  - hal\_freertos.c, 848
- ATCA\_NO\_DEVICES
  - atca\_status.h, 645
- ATCA\_NONCE
  - calib\_command.h, 726
- atca\_nonce\_in\_out, 431
- atca\_nonce\_in\_out\_t
  - Host side crypto methods (atcah\_), 316
- ATCA\_NOT\_INITIALIZED
  - atca\_status.h, 646
- ATCA\_NOT\_LOCKED
  - atca\_status.h, 645
- ATCA\_OPCODE\_IDX
  - calib\_command.h, 726
- atca\_openssl\_interface.c, 636
  - atcac\_aes\_cmac\_finish, 638
  - atcac\_aes\_cmac\_init, 638
  - atcac\_aes\_cmac\_update, 639
  - atcac\_aes\_gcm\_aad\_update, 639
  - atcac\_aes\_gcm\_decrypt\_finish, 640
  - atcac\_aes\_gcm\_decrypt\_start, 640
  - atcac\_aes\_gcm\_decrypt\_update, 640
  - atcac\_aes\_gcm\_encrypt\_finish, 641
  - atcac\_aes\_gcm\_encrypt\_start, 641
  - atcac\_aes\_gcm\_encrypt\_update, 642
  - atcac\_sw\_sha1\_finish, 642
  - atcac\_sw\_sha2\_256\_finish, 643
- ATCA\_OTP\_BLOCK\_MAX
  - calib\_command.h, 726
- ATCA\_OTP\_SIZE
  - calib\_command.h, 726
- ATCA\_P256\_KEY\_TYPE
  - calib\_command.h, 726
- ATCA\_PACKED
  - ATCADevice (atca\_), 134
  - Certificate manipulation methods (atcacert\_), 158
- ATCA\_PACKET\_OVERHEAD
  - calib\_command.h, 727
- ATCA\_PARAM1\_IDX
  - calib\_command.h, 727
- ATCA\_PARAM2\_IDX
  - calib\_command.h, 727
- ATCA\_PARITY\_ERROR
  - atca\_status.h, 645
- ATCA\_PARSE\_ERROR
  - atca\_status.h, 645
- ATCA\_PAUSE
  - calib\_command.h, 727
- ATCA\_POLLING\_FREQUENCY\_TIME\_MSEC
  - Hardware abstraction layer (hal\_), 262
- ATCA\_POLLING\_INIT\_TIME\_MSEC
  - Hardware abstraction layer (hal\_), 262
- ATCA\_POLLING\_MAX\_TIME\_MSEC
  - Hardware abstraction layer (hal\_), 262
- ATCA\_PRIV\_KEY\_SIZE
  - calib\_command.h, 727
- ATCA\_PRIVWRITE
  - calib\_command.h, 727
- ATCA\_PRIVWRITE\_MAC\_ZEROS\_SIZE
  - Host side crypto methods (atcah\_), 314
- ATCA\_PRIVWRITE\_PLAIN\_TEXT\_SIZE
  - Host side crypto methods (atcah\_), 314
- ATCA\_PUB\_KEY\_PAD

- calib\_command.h, 728
- ATCA\_PUB\_KEY\_SIZE
  - calib\_command.h, 728
- ATCA\_RANDOM
  - calib\_command.h, 728
- ATCA\_READ
  - calib\_command.h, 728
- ATCA\_RESYNC\_WITH\_WAKEUP
  - atca\_status.h, 645
- ATCA\_RSP\_DATA\_IDX
  - calib\_command.h, 728
- ATCA\_RSP\_SIZE\_16
  - calib\_command.h, 728
- ATCA\_RSP\_SIZE\_32
  - calib\_command.h, 729
- ATCA\_RSP\_SIZE\_4
  - calib\_command.h, 729
- ATCA\_RSP\_SIZE\_64
  - calib\_command.h, 729
- ATCA\_RSP\_SIZE\_72
  - calib\_command.h, 729
- ATCA\_RSP\_SIZE\_MAX
  - calib\_command.h, 729
- ATCA\_RSP\_SIZE\_MIN
  - calib\_command.h, 729
- ATCA\_RSP\_SIZE\_VAL
  - calib\_command.h, 730
- ATCA\_RX\_CRC\_ERROR
  - atca\_status.h, 645
- ATCA\_RX\_FAIL
  - atca\_status.h, 645
- ATCA\_RX\_NO\_RESPONSE
  - atca\_status.h, 645
- ATCA\_RX\_TIMEOUT
  - atca\_status.h, 645
- ATCA\_SECURE\_BOOT\_DIGEST
  - ATCADevice (atca\_), 135
- ATCA\_SECURE\_BOOT\_DIGEST\_MASK
  - ATCADevice (atca\_), 135
- ATCA\_SECURE\_BOOT\_DIGEST\_SHIFT
  - ATCADevice (atca\_), 135
- ATCA\_SECURE\_BOOT\_MODE
  - ATCADevice (atca\_), 135
- ATCA\_SECURE\_BOOT\_MODE\_MASK
  - ATCADevice (atca\_), 135
- ATCA\_SECURE\_BOOT\_MODE\_SHIFT
  - ATCADevice (atca\_), 135
- ATCA\_SECURE\_BOOT\_PERSIST\_EN\_MASK
  - ATCADevice (atca\_), 135
- ATCA\_SECURE\_BOOT\_PERSIST\_EN\_SHIFT
  - ATCADevice (atca\_), 136
- ATCA\_SECURE\_BOOT\_PUB\_KEY
  - ATCADevice (atca\_), 136
- ATCA\_SECURE\_BOOT\_PUB\_KEY\_MASK
  - ATCADevice (atca\_), 136
- ATCA\_SECURE\_BOOT\_PUB\_KEY\_SHIFT
  - ATCADevice (atca\_), 136
- ATCA\_SECURE\_BOOT\_RAND\_NONCE\_MASK
  - ATCADevice (atca\_), 136
- ATCA\_SECURE\_BOOT\_RAND\_NONCE\_SHIFT
  - ATCADevice (atca\_), 136
- ATCA\_SECUREBOOT
  - calib\_command.h, 730
- atca\_secureboot\_enc\_in\_out, 432
  - digest, 432
  - digest\_enc, 432
  - hashed\_key, 432
  - io\_key, 432
  - temp\_key, 433
- atca\_secureboot\_enc\_in\_out\_t
  - Host side crypto methods (atcah\_), 316
- atca\_secureboot\_mac\_in\_out, 433
  - digest, 433
  - hashed\_key, 433
  - mac, 434
  - mode, 434
  - param2, 434
  - secure\_boot\_config, 434
  - signature, 434
- atca\_secureboot\_mac\_in\_out\_t
  - Host side crypto methods (atcah\_), 316
- ATCA\_SELFTEST
  - calib\_command.h, 730
- ATCA\_SERIAL\_NUM\_SIZE
  - calib\_command.h, 730
- ATCA\_SHA
  - calib\_command.h, 730
- ATCA\_SHA1\_DIGEST\_SIZE
  - atca\_crypto\_sw.h, 580
- ATCA\_SHA206A\_DKEY\_CONSUMPTION\_MASK
  - api\_206a.h, 550
- ATCA\_SHA206A\_PKEY\_CONSUMPTION\_MASK
  - api\_206a.h, 550
- ATCA\_SHA206A\_SYMMETRIC\_KEY\_ID\_SLOT
  - api\_206a.h, 551
- ATCA\_SHA206A\_ZONE\_WRITE\_LOCK
  - api\_206a.h, 551
- ATCA\_SHA256\_BLOCK\_SIZE
  - cryptoauthlib.h, 824
- atca\_sha256\_ctx, 435
  - block, 435
  - block\_size, 435
  - total\_msg\_size, 435
- atca\_sha256\_ctx\_t
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 206
- ATCA\_SHA256\_DIGEST\_SIZE
  - cryptoauthlib.h, 825
- ATCA\_SHA256\_BLOCK\_SIZE
  - atca\_crypto\_sw.h, 580
- ATCA\_SHA256\_DIGEST\_SIZE
  - atca\_crypto\_sw.h, 581
- ATCA\_SHA\_CONFIG\_SIZE
  - calib\_command.h, 730
- ATCA\_SHA\_DIGEST\_SIZE
  - calib\_command.h, 731

ATCA\_SHA\_KEY\_TYPE  
     calib\_command.h, 731  
 ATCA\_SHA\_SUPPORT  
     cryptoauthlib.h, 825  
 ATCA\_SIG\_SIZE  
     calib\_command.h, 731  
 ATCA\_SIGN  
     calib\_command.h, 731  
 atca\_sign\_internal\_in\_out, 435  
     digest, 436  
     for\_invalidate, 436  
     is\_slot\_locked, 437  
     key\_config, 437  
     key\_id, 437  
     message, 437  
     mode, 437  
     slot\_config, 437  
     sn, 438  
     temp\_key, 438  
     update\_count, 438  
     use\_flag, 438  
     verify\_other\_data, 438  
 atca\_sign\_internal\_in\_out\_t  
     Host side crypto methods (atcah\_), 316  
 ATCA\_SLOT\_CONFIG\_ECDH\_MASK  
     ATCADevice (atca\_), 136  
 ATCA\_SLOT\_CONFIG\_ECDH\_SHIFT  
     ATCADevice (atca\_), 137  
 ATCA\_SLOT\_CONFIG\_ENCRYPTED\_READ\_MASK  
     ATCADevice (atca\_), 137  
 ATCA\_SLOT\_CONFIG\_ENCRYPTED\_READ\_SHIFT  
     ATCADevice (atca\_), 137  
 ATCA\_SLOT\_CONFIG\_EXT\_SIG\_MASK  
     ATCADevice (atca\_), 137  
 ATCA\_SLOT\_CONFIG\_EXT\_SIG\_SHIFT  
     ATCADevice (atca\_), 137  
 ATCA\_SLOT\_CONFIG\_GEN\_KEY\_MASK  
     ATCADevice (atca\_), 137  
 ATCA\_SLOT\_CONFIG\_GEN\_KEY\_SHIFT  
     ATCADevice (atca\_), 137  
 ATCA\_SLOT\_CONFIG\_INT\_SIG\_MASK  
     ATCADevice (atca\_), 137  
 ATCA\_SLOT\_CONFIG\_INT\_SIG\_SHIFT  
     ATCADevice (atca\_), 138  
 ATCA\_SLOT\_CONFIG\_IS\_SECRET\_MASK  
     ATCADevice (atca\_), 138  
 ATCA\_SLOT\_CONFIG\_IS\_SECRET\_SHIFT  
     ATCADevice (atca\_), 138  
 ATCA\_SLOT\_CONFIG\_LIMITED\_USE\_MASK  
     ATCADevice (atca\_), 138  
 ATCA\_SLOT\_CONFIG\_LIMITED\_USE\_SHIFT  
     ATCADevice (atca\_), 138  
 ATCA\_SLOT\_CONFIG\_NOMAC\_MASK  
     ATCADevice (atca\_), 138  
 ATCA\_SLOT\_CONFIG\_NOMAC\_SHIFT  
     ATCADevice (atca\_), 138  
 ATCA\_SLOT\_CONFIG\_PRIV\_WRITE\_MASK  
     ATCADevice (atca\_), 138  
 ATCA\_SLOT\_CONFIG\_PRIV\_WRITE\_SHIFT  
     ATCADevice (atca\_), 139  
 ATCA\_SLOT\_CONFIG\_READKEY  
     ATCADevice (atca\_), 139  
 ATCA\_SLOT\_CONFIG\_READKEY\_MASK  
     ATCADevice (atca\_), 139  
 ATCA\_SLOT\_CONFIG\_READKEY\_SHIFT  
     ATCADevice (atca\_), 139  
 ATCA\_SLOT\_CONFIG\_WRITE\_CONFIG  
     ATCADevice (atca\_), 139  
 ATCA\_SLOT\_CONFIG\_WRITE\_CONFIG\_MASK  
     ATCADevice (atca\_), 139  
 ATCA\_SLOT\_CONFIG\_WRITE\_CONFIG\_SHIFT  
     ATCADevice (atca\_), 139  
 ATCA\_SLOT\_CONFIG\_WRITE\_ECDH\_MASK  
     ATCADevice (atca\_), 140  
 ATCA\_SLOT\_CONFIG\_WRITE\_ECDH\_SHIFT  
     ATCADevice (atca\_), 140  
 ATCA\_SLOT\_CONFIG\_WRITE\_KEY  
     ATCADevice (atca\_), 140  
 ATCA\_SLOT\_CONFIG\_WRITE\_KEY\_MASK  
     ATCADevice (atca\_), 140  
 ATCA\_SLOT\_CONFIG\_WRITE\_KEY\_SHIFT  
     ATCADevice (atca\_), 140  
 ATCA\_SLOT\_LOCKED  
     ATCADevice (atca\_), 140  
 ATCA\_SMALL\_BUFFER  
     atca\_status.h, 645  
 ATCA\_SN\_0\_DEF  
     Host side crypto methods (atcah\_), 314  
 ATCA\_SN\_1\_DEF  
     Host side crypto methods (atcah\_), 314  
 ATCA\_SN\_8\_DEF  
     Host side crypto methods (atcah\_), 314  
 ATCA\_SPI\_IFACE  
     ATCAIface (atca\_), 147  
 atca\_start\_config.h, 643  
 atca\_start\_iface.h, 643  
 ATCA\_STATUS  
     atca\_status.h, 644  
 atca\_status.h, 643  
     ATCA\_ALLOC\_FAILURE, 646  
     ATCA\_ASSERT\_FAILURE, 645  
     ATCA\_BAD\_OPCODE, 645  
     ATCA\_BAD\_PARAM, 645  
     ATCA\_CHECKMAC\_VERIFY\_FAILED, 645  
     ATCA\_COMM\_FAIL, 645  
     ATCA\_CONFIG\_ZONE\_LOCKED, 645  
     ATCA\_DATA\_ZONE\_LOCKED, 645  
     ATCA\_EXECUTION\_ERROR, 645  
     ATCA\_FUNC\_FAIL, 645  
     ATCA\_GEN\_FAIL, 645  
     ATCA\_HEALTH\_TEST\_ERROR, 645  
     ATCA\_INVALID\_ID, 645  
     ATCA\_INVALID\_LENGTH, 645  
     ATCA\_INVALID\_POINTER, 645  
     ATCA\_INVALID\_SIZE, 645  
     ATCA\_NO\_DEVICES, 645

- ATCA\_NOT\_INITIALIZED, [646](#)
- ATCA\_NOT\_LOCKED, [645](#)
- ATCA\_PARITY\_ERROR, [645](#)
- ATCA\_PARSE\_ERROR, [645](#)
- ATCA\_RESYNC\_WITH\_WAKEUP, [645](#)
- ATCA\_RX\_CRC\_ERROR, [645](#)
- ATCA\_RX\_FAIL, [645](#)
- ATCA\_RX\_NO\_RESPONSE, [645](#)
- ATCA\_RX\_TIMEOUT, [645](#)
- ATCA\_SMALL\_BUFFER, [645](#)
- ATCA\_STATUS, [644](#)
- ATCA\_STATUS\_AUTH\_BIT, [644](#)
- ATCA\_STATUS\_CRC, [645](#)
- ATCA\_STATUS\_ECC, [645](#)
- ATCA\_STATUS\_SELFTEST\_ERROR, [645](#)
- ATCA\_STATUS\_UNKNOWN, [645](#)
- ATCA\_SUCCESS, [645](#)
- ATCA\_TIMEOUT, [645](#)
- ATCA\_TOO\_MANY\_COMM\_RETRIES, [645](#)
- ATCA\_TX\_FAIL, [645](#)
- ATCA\_TX\_TIMEOUT, [645](#)
- ATCA\_UNIMPLEMENTED, [645](#)
- ATCA\_USE\_FLAGS\_CONSUMED, [646](#)
- ATCA\_WAKE\_FAILED, [645](#)
- ATCA\_WAKE\_SUCCESS, [645](#)
- ATCA\_STATUS\_AUTH\_BIT
  - atca\_status.h, [644](#)
- ATCA\_STATUS\_CRC
  - atca\_status.h, [645](#)
- ATCA\_STATUS\_ECC
  - atca\_status.h, [645](#)
- ATCA\_STATUS\_SELFTEST\_ERROR
  - atca\_status.h, [645](#)
- ATCA\_STATUS\_UNKNOWN
  - atca\_status.h, [645](#)
- ATCA\_STRINGIFY
  - cryptoauthlib.h, [825](#)
- ATCA\_SUCCESS
  - atca\_status.h, [645](#)
- ATCA\_SWI\_IFACE
  - ATCAIface (atca\_), [147](#)
- ATCA\_TA\_SUPPORT
  - cryptoauthlib.h, [825](#)
- atca\_temp\_key, [439](#)
  - gen\_dig\_data, [439](#)
  - gen\_key\_data, [439](#)
  - is\_64, [439](#)
  - key\_id, [440](#)
  - no\_mac\_flag, [440](#)
  - source\_flag, [440](#)
  - valid, [440](#)
  - value, [440](#)
- atca\_temp\_key\_t
  - Host side crypto methods (atcah\_), [316](#)
- ATCA\_TEMPKEY\_KEYID
  - calib\_command.h, [731](#)
- ATCA\_TIMEOUT
  - atca\_status.h, [645](#)
- ATCA\_TOO\_MANY\_COMM\_RETRIES
  - atca\_status.h, [645](#)
- ATCA\_TOSTRING
  - cryptoauthlib.h, [825](#)
- ATCA\_TRACE
  - cryptoauthlib.h, [825](#)
- atca\_trace
  - atca\_debug.c, [590](#)
  - atca\_debug.h, [591](#)
- atca\_trace\_config
  - atca\_debug.c, [590](#)
  - atca\_debug.h, [591](#)
- atca\_trace\_msg
  - atca\_debug.c, [590](#)
  - atca\_debug.h, [591](#)
- ATCA\_TX\_FAIL
  - atca\_status.h, [645](#)
- ATCA\_TX\_TIMEOUT
  - atca\_status.h, [645](#)
- ATCA\_UART\_IFACE
  - ATCAIface (atca\_), [147](#)
- ATCA\_UNIMPLEMENTED
  - atca\_status.h, [645](#)
- ATCA\_UNKNOWN\_IFACE
  - ATCAIface (atca\_), [147](#)
- ATCA\_UNLOCKED
  - calib\_command.h, [731](#)
- ATCA\_UNSUPPORTED\_CMD
  - calib\_execution.h, [805](#)
- ATCA\_UPDATE\_EXTRA
  - calib\_command.h, [732](#)
- ATCA\_USE\_FLAGS\_CONSUMED
  - atca\_status.h, [646](#)
- ATCA\_USE\_LOCK\_ENABLE\_MASK
  - ATCADevice (atca\_), [140](#)
- ATCA\_USE\_LOCK\_ENABLE\_SHIFT
  - ATCADevice (atca\_), [141](#)
- ATCA\_USE\_LOCK\_KEY\_MASK
  - ATCADevice (atca\_), [141](#)
- ATCA\_USE\_LOCK\_KEY\_SHIFT
  - ATCADevice (atca\_), [141](#)
- ATCA\_VERIFY
  - calib\_command.h, [732](#)
- atca\_verify\_in\_out, [441](#)
- atca\_verify\_in\_out\_t
  - Host side crypto methods (atcah\_), [317](#)
- atca\_verify\_mac, [441](#)
  - io\_key, [442](#)
  - key\_id, [442](#)
  - mac, [442](#)
  - mode, [442](#)
  - msg\_dig\_buf, [442](#)
  - other\_data, [442](#)
  - signature, [443](#)
  - sn, [443](#)
  - temp\_key, [443](#)
- atca\_verify\_mac\_in\_out\_t
  - Host side crypto methods (atcah\_), [317](#)

atca\_version  
     atca\_basic.c, 563  
 atca\_version.h, 646  
     ATCA\_LIBRARY\_VERSION\_BUILD, 646  
     ATCA\_LIBRARY\_VERSION\_DATE, 646  
     ATCA\_LIBRARY\_VERSION\_MAJOR, 646  
     ATCA\_LIBRARY\_VERSION\_MINOR, 647  
 ATCA\_VOL\_KEY\_PERM\_EN\_MASK  
     ATCADevice (atca\_), 141  
 ATCA\_VOL\_KEY\_PERM\_EN\_SHIFT  
     ATCADevice (atca\_), 141  
 ATCA\_VOL\_KEY\_PERM\_SLOT  
     ATCADevice (atca\_), 141  
 ATCA\_VOL\_KEY\_PERM\_SLOT\_MASK  
     ATCADevice (atca\_), 141  
 ATCA\_VOL\_KEY\_PERM\_SLOT\_SHIFT  
     ATCADevice (atca\_), 142  
 ATCA\_WAKE\_FAILED  
     atca\_status.h, 645  
 ATCA\_WAKE\_SUCCESS  
     atca\_status.h, 645  
 atca\_wolfssl\_interface.c, 647  
 ATCA\_WORD\_SIZE  
     calib\_command.h, 732  
 ATCA\_WRITE  
     calib\_command.h, 732  
 atca\_write\_mac\_in\_out, 443  
     auth\_mac, 444  
     encrypted\_data, 444  
     input\_data, 444  
     key\_id, 444  
     sn, 444  
     temp\_key, 444  
     zone, 445  
 atca\_write\_mac\_in\_out\_t  
     Host side crypto methods (atcah\_), 317  
 ATCA\_WRITE\_MAC\_ZEROS\_SIZE  
     Host side crypto methods (atcah\_), 315  
 ATCA\_ZONE\_CONFIG  
     cryptoauthlib.h, 825  
 ATCA\_ZONE\_DATA  
     cryptoauthlib.h, 826  
 ATCA\_ZONE\_ENCRYPTED  
     calib\_command.h, 732  
 ATCA\_ZONE\_MASK  
     calib\_command.h, 732  
 ATCA\_ZONE\_OTP  
     cryptoauthlib.h, 826  
 ATCA\_ZONE\_READWRITE\_32  
     calib\_command.h, 733  
 atcab\_aes  
     Basic Crypto API methods (atcab\_), 60  
 atcab\_aes\_cbc\_decrypt\_block  
     Basic Crypto API methods (atcab\_), 60  
 atcab\_aes\_cbc\_encrypt\_block  
     Basic Crypto API methods (atcab\_), 61  
 atcab\_aes\_cbc\_encrypt\_block\_ext  
     Basic Crypto API methods (atcab\_), 61  
 atcab\_aes\_cbc\_init  
     Basic Crypto API methods (atcab\_), 61  
 atcab\_aes\_cbc\_init\_ext  
     Basic Crypto API methods (atcab\_), 62  
 atcab\_aes\_cmac\_finish  
     Basic Crypto API methods (atcab\_), 62  
 atcab\_aes\_cmac\_init  
     Basic Crypto API methods (atcab\_), 63  
 atcab\_aes\_cmac\_init\_ext  
     Basic Crypto API methods (atcab\_), 63  
 atcab\_aes\_cmac\_update  
     Basic Crypto API methods (atcab\_), 64  
 atcab\_aes\_ctr\_block  
     Basic Crypto API methods (atcab\_), 64  
 atcab\_aes\_ctr\_decrypt\_block  
     Basic Crypto API methods (atcab\_), 64  
 atcab\_aes\_ctr\_encrypt\_block  
     Basic Crypto API methods (atcab\_), 65  
 atcab\_aes\_ctr\_increment  
     Basic Crypto API methods (atcab\_), 65  
 atcab\_aes\_ctr\_init  
     Basic Crypto API methods (atcab\_), 66  
 atcab\_aes\_ctr\_init\_ext  
     Basic Crypto API methods (atcab\_), 66  
 atcab\_aes\_ctr\_init\_rand  
     Basic Crypto API methods (atcab\_), 67  
 atcab\_aes\_ctr\_init\_rand\_ext  
     Basic Crypto API methods (atcab\_), 67  
 atcab\_aes\_decrypt  
     Basic Crypto API methods (atcab\_), 68  
 atcab\_aes\_decrypt\_ext  
     Basic Crypto API methods (atcab\_), 68  
 atcab\_aes\_encrypt  
     Basic Crypto API methods (atcab\_), 69  
 atcab\_aes\_encrypt\_ext  
     Basic Crypto API methods (atcab\_), 69  
 atcab\_aes\_gcm\_aad\_update  
     Basic Crypto API methods (atcab\_), 70  
 atcab\_aes\_gcm\_decrypt\_finish  
     Basic Crypto API methods (atcab\_), 70  
 atcab\_aes\_gcm\_decrypt\_update  
     Basic Crypto API methods (atcab\_), 71  
 atcab\_aes\_gcm\_encrypt\_finish  
     Basic Crypto API methods (atcab\_), 71  
 atcab\_aes\_gcm\_encrypt\_update  
     Basic Crypto API methods (atcab\_), 72  
 atcab\_aes\_gcm\_init  
     Basic Crypto API methods (atcab\_), 72  
 atcab\_aes\_gcm\_init\_rand  
     Basic Crypto API methods (atcab\_), 73  
 atcab\_aes\_gfm  
     Basic Crypto API methods (atcab\_), 73  
 atcab\_b64rules\_default  
     atca\_helpers.c, 609  
     atca\_helpers.h, 619  
 atcab\_b64rules\_mime  
     atca\_helpers.c, 609  
     atca\_helpers.h, 619

- atcab\_b64rules\_urlsafe
  - atca\_helpers.c, [609](#)
  - atca\_helpers.h, [619](#)
- atcab\_base64decode
  - atca\_helpers.c, [599](#)
  - atca\_helpers.h, [610](#)
- atcab\_base64decode\_
  - atca\_helpers.c, [600](#)
  - atca\_helpers.h, [611](#)
- atcab\_base64encode
  - atca\_helpers.c, [600](#)
  - atca\_helpers.h, [611](#)
- atcab\_base64encode\_
  - atca\_helpers.c, [601](#)
  - atca\_helpers.h, [612](#)
- atcab\_bin2hex
  - atca\_helpers.c, [601](#)
  - atca\_helpers.h, [612](#)
- atcab\_bin2hex\_
  - atca\_helpers.c, [601](#)
  - atca\_helpers.h, [613](#)
- atcab\_cfg\_discover
  - Basic Crypto API methods (atcab\_), [59](#)
- atcab\_challenge
  - Basic Crypto API methods (atcab\_), [74](#)
- atcab\_challenge\_seed\_update
  - Basic Crypto API methods (atcab\_), [74](#)
- atcab\_checkmac
  - Basic Crypto API methods (atcab\_), [75](#)
- atcab\_cmp\_config\_zone
  - Basic Crypto API methods (atcab\_), [75](#)
- atcab\_counter
  - Basic Crypto API methods (atcab\_), [76](#)
- atcab\_counter\_increment
  - Basic Crypto API methods (atcab\_), [76](#)
- atcab\_counter\_read
  - Basic Crypto API methods (atcab\_), [76](#)
- atcab\_derivekey
  - Basic Crypto API methods (atcab\_), [77](#)
- atcab\_ecdh
  - Basic Crypto API methods (atcab\_), [77](#)
- atcab\_ecdh\_base
  - Basic Crypto API methods (atcab\_), [78](#)
- atcab\_ecdh\_enc
  - Basic Crypto API methods (atcab\_), [78](#)
- atcab\_ecdh\_ioenc
  - Basic Crypto API methods (atcab\_), [79](#)
- atcab\_ecdh\_tempkey
  - Basic Crypto API methods (atcab\_), [79](#)
- atcab\_ecdh\_tempkey\_ioenc
  - Basic Crypto API methods (atcab\_), [80](#)
- atcab\_gendig
  - Basic Crypto API methods (atcab\_), [80](#)
- atcab\_genkey
  - Basic Crypto API methods (atcab\_), [81](#)
- atcab\_genkey\_base
  - Basic Crypto API methods (atcab\_), [81](#)
- atcab\_get\_addr
  - Basic Crypto API methods (atcab\_), [59](#)
- atcab\_get\_device
  - Basic Crypto API methods (atcab\_), [82](#)
- atcab\_get\_device\_type
  - Basic Crypto API methods (atcab\_), [82](#)
- atcab\_get\_device\_type\_ext
  - Basic Crypto API methods (atcab\_), [82](#)
- atcab\_get\_pubkey
  - Basic Crypto API methods (atcab\_), [82](#)
- atcab\_get\_zone\_size
  - Basic Crypto API methods (atcab\_), [83](#)
- atcab\_hex2bin
  - atca\_helpers.c, [602](#)
  - atca\_helpers.h, [613](#)
- atcab\_hex2bin\_
  - atca\_helpers.c, [602](#)
  - atca\_helpers.h, [614](#)
- atcab\_hmac
  - Basic Crypto API methods (atcab\_), [83](#)
- atcab\_hw\_sha2\_256
  - Basic Crypto API methods (atcab\_), [84](#)
- atcab\_hw\_sha2\_256\_finish
  - Basic Crypto API methods (atcab\_), [84](#)
- atcab\_hw\_sha2\_256\_init
  - Basic Crypto API methods (atcab\_), [84](#)
- atcab\_hw\_sha2\_256\_update
  - Basic Crypto API methods (atcab\_), [85](#)
- atcab\_idle
  - Basic Crypto API methods (atcab\_), [85](#)
- atcab\_info
  - Basic Crypto API methods (atcab\_), [85](#)
- atcab\_info\_base
  - Basic Crypto API methods (atcab\_), [86](#)
- atcab\_info\_get\_latch
  - Basic Crypto API methods (atcab\_), [86](#)
- atcab\_info\_set\_latch
  - Basic Crypto API methods (atcab\_), [86](#)
- atcab\_init
  - Basic Crypto API methods (atcab\_), [87](#)
- atcab\_init\_device
  - Basic Crypto API methods (atcab\_), [87](#)
- atcab\_init\_ext
  - Basic Crypto API methods (atcab\_), [88](#)
- atcab\_is\_ca\_device
  - Basic Crypto API methods (atcab\_), [88](#)
- atcab\_is\_config\_locked
  - Basic Crypto API methods (atcab\_), [88](#)
- atcab\_is\_data\_locked
  - Basic Crypto API methods (atcab\_), [89](#)
- atcab\_is\_locked
  - Basic Crypto API methods (atcab\_), [89](#)
- atcab\_is\_slot\_locked
  - Basic Crypto API methods (atcab\_), [89](#)
- atcab\_is\_ta\_device
  - Basic Crypto API methods (atcab\_), [90](#)
- atcab\_kdf
  - Basic Crypto API methods (atcab\_), [90](#)
- atcab\_lock



Basic Crypto API methods (atcab\_), 91  
 atcab\_lock\_config\_zone  
     Basic Crypto API methods (atcab\_), 91  
 atcab\_lock\_config\_zone\_crc  
     Basic Crypto API methods (atcab\_), 91  
 atcab\_lock\_data\_slot  
     Basic Crypto API methods (atcab\_), 92  
 atcab\_lock\_data\_zone  
     Basic Crypto API methods (atcab\_), 92  
 atcab\_lock\_data\_zone\_crc  
     Basic Crypto API methods (atcab\_), 92  
 atcab\_mac  
     Basic Crypto API methods (atcab\_), 93  
 atcab\_memset\_s  
     atca\_helpers.c, 603  
     atca\_helpers.h, 614  
 atcab\_nonce  
     Basic Crypto API methods (atcab\_), 93  
 atcab\_nonce\_base  
     Basic Crypto API methods (atcab\_), 94  
 atcab\_nonce\_load  
     Basic Crypto API methods (atcab\_), 94  
 atcab\_nonce\_rand  
     Basic Crypto API methods (atcab\_), 95  
 atcab\_printbin  
     Basic Crypto API methods (atcab\_), 95  
 atcab\_printbin\_label  
     atca\_helpers.h, 614  
 atcab\_printbin\_sp  
     atca\_helpers.h, 614  
 atcab\_priv\_write  
     Basic Crypto API methods (atcab\_), 95  
 atcab\_random  
     Basic Crypto API methods (atcab\_), 96  
 atcab\_random\_ext  
     Basic Crypto API methods (atcab\_), 96  
 atcab\_read\_bytes\_zone  
     Basic Crypto API methods (atcab\_), 96  
 atcab\_read\_config\_zone  
     Basic Crypto API methods (atcab\_), 97  
 atcab\_read\_enc  
     Basic Crypto API methods (atcab\_), 97  
 atcab\_read\_pubkey  
     Basic Crypto API methods (atcab\_), 98  
 atcab\_read\_serial\_number  
     Basic Crypto API methods (atcab\_), 98  
 atcab\_read\_sig  
     Basic Crypto API methods (atcab\_), 99  
 atcab\_read\_zone  
     Basic Crypto API methods (atcab\_), 99  
 atcab\_release  
     Basic Crypto API methods (atcab\_), 99  
 atcab\_release\_ext  
     Basic Crypto API methods (atcab\_), 100  
 atcab\_reversal  
     atca\_helpers.c, 603  
     atca\_helpers.h, 614  
 atcab\_secureboot

Basic Crypto API methods (atcab\_), 100  
 atcab\_secureboot\_mac  
     Basic Crypto API methods (atcab\_), 101  
 atcab\_selftest  
     Basic Crypto API methods (atcab\_), 101  
 atcab\_sha  
     Basic Crypto API methods (atcab\_), 102  
 atcab\_sha\_base  
     Basic Crypto API methods (atcab\_), 102  
 atcab\_sha\_end  
     Basic Crypto API methods (atcab\_), 103  
 atcab\_sha\_hmac  
     Basic Crypto API methods (atcab\_), 103  
 atcab\_sha\_hmac\_finish  
     Basic Crypto API methods (atcab\_), 104  
 atcab\_sha\_hmac\_init  
     Basic Crypto API methods (atcab\_), 104  
 atcab\_sha\_hmac\_update  
     Basic Crypto API methods (atcab\_), 104  
 atcab\_sha\_read\_context  
     Basic Crypto API methods (atcab\_), 105  
 atcab\_sha\_start  
     Basic Crypto API methods (atcab\_), 105  
 atcab\_sha\_update  
     Basic Crypto API methods (atcab\_), 105  
 atcab\_sha\_write\_context  
     Basic Crypto API methods (atcab\_), 106  
 atcab\_sign  
     Basic Crypto API methods (atcab\_), 106  
 atcab\_sign\_base  
     Basic Crypto API methods (atcab\_), 107  
 atcab\_sign\_internal  
     Basic Crypto API methods (atcab\_), 107  
 atcab\_sleep  
     Basic Crypto API methods (atcab\_), 108  
 atcab\_updateextra  
     Basic Crypto API methods (atcab\_), 108  
 atcab\_verify  
     Basic Crypto API methods (atcab\_), 108  
 atcab\_verify\_extern  
     Basic Crypto API methods (atcab\_), 109  
 atcab\_verify\_extern\_mac  
     Basic Crypto API methods (atcab\_), 110  
 atcab\_verify\_invalidate  
     Basic Crypto API methods (atcab\_), 110  
 atcab\_verify\_stored  
     Basic Crypto API methods (atcab\_), 111  
 atcab\_verify\_stored\_mac  
     Basic Crypto API methods (atcab\_), 111  
 atcab\_verify\_validate  
     Basic Crypto API methods (atcab\_), 112  
 atcab\_version  
     Basic Crypto API methods (atcab\_), 112  
 atcab\_wakeup  
     Basic Crypto API methods (atcab\_), 113  
 atcab\_write  
     Basic Crypto API methods (atcab\_), 113  
 atcab\_write\_bytes\_zone

- Basic Crypto API methods (atcab\_), 113
- atcab\_write\_config\_counter
  - Basic Crypto API methods (atcab\_), 114
- atcab\_write\_config\_zone
  - Basic Crypto API methods (atcab\_), 114
- atcab\_write\_enc
  - Basic Crypto API methods (atcab\_), 115
- atcab\_write\_pubkey
  - Basic Crypto API methods (atcab\_), 115
- atcab\_write\_zone
  - Basic Crypto API methods (atcab\_), 116
- atcac\_aes\_cmac\_ctx
  - atca\_crypto\_sw.h, 581
- atcac\_aes\_cmac\_finish
  - atca\_crypto\_sw.h, 582
  - atca\_mbedtls\_wrap.c, 631
  - atca\_openssl\_interface.c, 638
- atcac\_aes\_cmac\_init
  - atca\_crypto\_sw.h, 582
  - atca\_mbedtls\_wrap.c, 631
  - atca\_openssl\_interface.c, 638
- atcac\_aes\_cmac\_update
  - atca\_crypto\_sw.h, 582
  - atca\_mbedtls\_wrap.c, 632
  - atca\_openssl\_interface.c, 639
- atcac\_aes\_gcm\_aad\_update
  - atca\_crypto\_sw.h, 582
  - atca\_mbedtls\_wrap.c, 632
  - atca\_openssl\_interface.c, 639
- atcac\_aes\_gcm\_ctx
  - atca\_crypto\_sw.h, 581
- atcac\_aes\_gcm\_decrypt\_finish
  - atca\_crypto\_sw.h, 583
  - atca\_mbedtls\_wrap.c, 632
  - atca\_openssl\_interface.c, 640
- atcac\_aes\_gcm\_decrypt\_start
  - atca\_crypto\_sw.h, 583
  - atca\_mbedtls\_wrap.c, 633
  - atca\_openssl\_interface.c, 640
- atcac\_aes\_gcm\_decrypt\_update
  - atca\_crypto\_sw.h, 583
  - atca\_mbedtls\_wrap.c, 633
  - atca\_openssl\_interface.c, 640
- atcac\_aes\_gcm\_encrypt\_finish
  - atca\_crypto\_sw.h, 584
  - atca\_mbedtls\_wrap.c, 634
  - atca\_openssl\_interface.c, 641
- atcac\_aes\_gcm\_encrypt\_start
  - atca\_crypto\_sw.h, 584
  - atca\_mbedtls\_wrap.c, 634
  - atca\_openssl\_interface.c, 641
- atcac\_aes\_gcm\_encrypt\_update
  - atca\_crypto\_sw.h, 584
  - atca\_mbedtls\_wrap.c, 635
  - atca\_openssl\_interface.c, 642
- atcac\_hmac\_sha256\_ctx
  - atca\_crypto\_sw.h, 581
- atcac\_sha1\_ctx
  - atca\_crypto\_sw.h, 581
- atcac\_sha256\_hmac\_finish
  - Software crypto methods (atcac\_), 252
- atcac\_sha256\_hmac\_init
  - Software crypto methods (atcac\_), 252
- atcac\_sha256\_hmac\_update
  - Software crypto methods (atcac\_), 253
- atcac\_sha2\_256\_ctx
  - atca\_crypto\_sw.h, 581
- atcac\_sw\_ecdsa\_verify\_p256
  - Software crypto methods (atcac\_), 253
- atcac\_sw\_random
  - Software crypto methods (atcac\_), 253
- atcac\_sw\_sha1
  - Software crypto methods (atcac\_), 254
- atcac\_sw\_sha1\_finish
  - atca\_mbedtls\_wrap.c, 635
  - atca\_openssl\_interface.c, 642
  - Software crypto methods (atcac\_), 254
- atcac\_sw\_sha1\_init
  - Software crypto methods (atcac\_), 254
- atcac\_sw\_sha1\_update
  - Software crypto methods (atcac\_), 255
- atcac\_sw\_sha2\_256
  - Software crypto methods (atcac\_), 255
- atcac\_sw\_sha2\_256\_finish
  - atca\_mbedtls\_wrap.c, 636
  - atca\_openssl\_interface.c, 643
  - Software crypto methods (atcac\_), 255
- atcac\_sw\_sha2\_256\_init
  - Software crypto methods (atcac\_), 255
- atcac\_sw\_sha2\_256\_update
  - Software crypto methods (atcac\_), 256
- atcacert.h, 647
- atcacert\_build\_state\_s, 445
  - cert, 445
  - cert\_def, 446
  - cert\_size, 446
  - device\_sn, 446
  - is\_device\_sn, 446
  - max\_cert\_size, 446
- atcacert\_build\_state\_t
  - Certificate manipulation methods (atcacert\_), 162
- atcacert\_cert\_build\_finish
  - Certificate manipulation methods (atcacert\_), 168
- atcacert\_cert\_build\_process
  - Certificate manipulation methods (atcacert\_), 168
- atcacert\_cert\_build\_start
  - Certificate manipulation methods (atcacert\_), 169
- atcacert\_cert\_element\_s, 447
  - cert\_loc, 447
  - device\_loc, 447
  - id, 447
  - transforms, 447
- atcacert\_cert\_element\_t
  - Certificate manipulation methods (atcacert\_), 162
- atcacert\_cert\_loc\_s, 448
  - count, 448



- offset, [448](#)
- atcacert\_cert\_loc\_t
  - Certificate manipulation methods (atcacert\_), [162](#)
- atcacert\_cert\_sn\_src\_e
  - Certificate manipulation methods (atcacert\_), [164](#)
- atcacert\_cert\_sn\_src\_t
  - Certificate manipulation methods (atcacert\_), [162](#)
- atcacert\_cert\_type\_e
  - Certificate manipulation methods (atcacert\_), [165](#)
- atcacert\_cert\_type\_t
  - Certificate manipulation methods (atcacert\_), [162](#)
- atcacert\_client.c, [648](#)
- atcacert\_client.h, [649](#)
- atcacert\_create\_csr
  - Certificate manipulation methods (atcacert\_), [169](#)
- atcacert\_create\_csr\_pem
  - Certificate manipulation methods (atcacert\_), [170](#)
- atcacert\_date.c, [650](#)
- atcacert\_date.h, [651](#)
- atcacert\_date\_dec
  - Certificate manipulation methods (atcacert\_), [170](#)
- atcacert\_date\_dec\_compcert
  - Certificate manipulation methods (atcacert\_), [171](#)
- atcacert\_date\_dec\_iso8601\_sep
  - Certificate manipulation methods (atcacert\_), [171](#)
- atcacert\_date\_dec\_posix\_uint32\_be
  - Certificate manipulation methods (atcacert\_), [171](#)
- atcacert\_date\_dec\_posix\_uint32\_le
  - Certificate manipulation methods (atcacert\_), [172](#)
- atcacert\_date\_dec\_rfc5280\_gen
  - Certificate manipulation methods (atcacert\_), [172](#)
- atcacert\_date\_dec\_rfc5280\_utc
  - Certificate manipulation methods (atcacert\_), [172](#)
- atcacert\_date\_enc
  - Certificate manipulation methods (atcacert\_), [172](#)
- atcacert\_date\_enc\_compcert
  - Certificate manipulation methods (atcacert\_), [173](#)
- atcacert\_date\_enc\_iso8601\_sep
  - Certificate manipulation methods (atcacert\_), [173](#)
- atcacert\_date\_enc\_posix\_uint32\_be
  - Certificate manipulation methods (atcacert\_), [173](#)
- atcacert\_date\_enc\_posix\_uint32\_le
  - Certificate manipulation methods (atcacert\_), [173](#)
- atcacert\_date\_enc\_rfc5280\_gen
  - Certificate manipulation methods (atcacert\_), [173](#)
- atcacert\_date\_enc\_rfc5280\_utc
  - Certificate manipulation methods (atcacert\_), [174](#)
- atcacert\_date\_format\_e
  - Certificate manipulation methods (atcacert\_), [165](#)
- ATCACERT\_DATE\_FORMAT\_SIZES
  - Certificate manipulation methods (atcacert\_), [200](#)
- ATCACERT\_DATE\_FORMAT\_SIZES\_COUNT
  - Certificate manipulation methods (atcacert\_), [158](#)
- atcacert\_date\_format\_t
  - Certificate manipulation methods (atcacert\_), [163](#)
- atcacert\_date\_get\_max\_date
  - Certificate manipulation methods (atcacert\_), [174](#)
- atcacert\_decode\_pem
  - atcacert\_pem.c, [664](#)
  - atcacert\_pem.h, [669](#)
- atcacert\_decode\_pem\_cert
  - atcacert\_pem.c, [665](#)
  - atcacert\_pem.h, [669](#)
- atcacert\_decode\_pem\_csr
  - atcacert\_pem.c, [665](#)
  - atcacert\_pem.h, [670](#)
- atcacert\_def.c, [653](#)
- ATCACERT\_MAX, [655](#)
- ATCACERT\_MIN, [656](#)
- atcacert\_def.h, [656](#)
- ATCA\_MAX\_TRANSFORMS, [659](#)
- atcacert\_def\_s, [448](#)
- ca\_cert\_def, [449](#)
- cert\_elements, [450](#)
- cert\_elements\_count, [450](#)
- cert\_sn\_dev\_loc, [450](#)
- cert\_template, [450](#)
- cert\_template\_size, [450](#)
- chain\_id, [450](#)
- comp\_cert\_dev\_loc, [451](#)
- expire\_date\_format, [451](#)
- expire\_years, [451](#)
- issue\_date\_format, [451](#)
- private\_key\_slot, [451](#)
- public\_key\_dev\_loc, [451](#)
- sn\_source, [452](#)
- std\_cert\_elements, [452](#)
- tbs\_cert\_loc, [452](#)
- template\_id, [452](#)
- type, [452](#)
- atcacert\_def\_t
  - Certificate manipulation methods (atcacert\_), [163](#)
- atcacert\_der.c, [660](#)
- atcacert\_der.h, [660](#)
- atcacert\_der\_adjust\_length
  - Certificate manipulation methods (atcacert\_), [174](#)
- atcacert\_der\_dec\_ecdsa\_sig\_value
  - Certificate manipulation methods (atcacert\_), [174](#)
- atcacert\_der\_dec\_integer
  - Certificate manipulation methods (atcacert\_), [175](#)
- atcacert\_der\_dec\_length
  - Certificate manipulation methods (atcacert\_), [176](#)
- atcacert\_der\_enc\_ecdsa\_sig\_value
  - Certificate manipulation methods (atcacert\_), [176](#)
- atcacert\_der\_enc\_integer
  - Certificate manipulation methods (atcacert\_), [177](#)
- atcacert\_der\_enc\_length
  - Certificate manipulation methods (atcacert\_), [177](#)
- atcacert\_device\_loc\_s, [453](#)
- count, [453](#)
- is\_genkey, [453](#)
- offset, [453](#)
- slot, [453](#)
- zone, [454](#)
- atcacert\_device\_loc\_t
  - Certificate manipulation methods (atcacert\_), [163](#)

- atcacert\_device\_zone\_e
  - Certificate manipulation methods (atcacert\_), 166
- atcacert\_device\_zone\_t
  - Certificate manipulation methods (atcacert\_), 163
- ATCACERT\_E\_BAD\_CERT
  - Certificate manipulation methods (atcacert\_), 158
- ATCACERT\_E\_BAD\_PARAMS
  - Certificate manipulation methods (atcacert\_), 159
- ATCACERT\_E\_BUFFER\_TOO\_SMALL
  - Certificate manipulation methods (atcacert\_), 159
- ATCACERT\_E\_DECODING\_ERROR
  - Certificate manipulation methods (atcacert\_), 159
- ATCACERT\_E\_ELEM\_MISSING
  - Certificate manipulation methods (atcacert\_), 159
- ATCACERT\_E\_ELEM\_OUT\_OF\_BOUNDS
  - Certificate manipulation methods (atcacert\_), 159
- ATCACERT\_E\_ERROR
  - Certificate manipulation methods (atcacert\_), 159
- ATCACERT\_E\_INVALID\_DATE
  - Certificate manipulation methods (atcacert\_), 160
- ATCACERT\_E\_INVALID\_TRANSFORM
  - Certificate manipulation methods (atcacert\_), 160
- ATCACERT\_E\_SUCCESS
  - Certificate manipulation methods (atcacert\_), 160
- ATCACERT\_E\_UNEXPECTED\_ELEM\_SIZE
  - Certificate manipulation methods (atcacert\_), 160
- ATCACERT\_E\_UNIMPLEMENTED
  - Certificate manipulation methods (atcacert\_), 160
- ATCACERT\_E\_VERIFY\_FAILED
  - Certificate manipulation methods (atcacert\_), 160
- ATCACERT\_E\_WRONG\_CERT\_DEF
  - Certificate manipulation methods (atcacert\_), 161
- atcacert\_encode\_pem
  - atcacert\_pem.c, 666
  - atcacert\_pem.h, 670
- atcacert\_encode\_pem\_cert
  - atcacert\_pem.c, 666
  - atcacert\_pem.h, 671
- atcacert\_encode\_pem\_csr
  - atcacert\_pem.c, 667
  - atcacert\_pem.h, 671
- atcacert\_gen\_cert\_sn
  - Certificate manipulation methods (atcacert\_), 178
- atcacert\_gen\_challenge\_hw
  - Certificate manipulation methods (atcacert\_), 178
- atcacert\_gen\_challenge\_sw
  - Certificate manipulation methods (atcacert\_), 179
- atcacert\_get\_auth\_key\_id
  - Certificate manipulation methods (atcacert\_), 179
- atcacert\_get\_cert\_element
  - Certificate manipulation methods (atcacert\_), 179
- atcacert\_get\_cert\_sn
  - Certificate manipulation methods (atcacert\_), 180
- atcacert\_get\_comp\_cert
  - Certificate manipulation methods (atcacert\_), 180
- atcacert\_get\_device\_data
  - Certificate manipulation methods (atcacert\_), 181
- atcacert\_get\_device\_locs
  - Certificate manipulation methods (atcacert\_), 181
- atcacert\_get\_expire\_date
  - Certificate manipulation methods (atcacert\_), 182
- atcacert\_get\_issue\_date
  - Certificate manipulation methods (atcacert\_), 183
- atcacert\_get\_key\_id
  - Certificate manipulation methods (atcacert\_), 183
- atcacert\_get\_response
  - Certificate manipulation methods (atcacert\_), 184
- atcacert\_get\_signature
  - Certificate manipulation methods (atcacert\_), 184
- atcacert\_get\_signer\_id
  - Certificate manipulation methods (atcacert\_), 185
- atcacert\_get\_subj\_key\_id
  - Certificate manipulation methods (atcacert\_), 185
- atcacert\_get\_subj\_public\_key
  - Certificate manipulation methods (atcacert\_), 186
- atcacert\_get\_tbs
  - Certificate manipulation methods (atcacert\_), 186
- atcacert\_get\_tbs\_digest
  - Certificate manipulation methods (atcacert\_), 187
- atcacert\_host\_hw.c, 661
- atcacert\_host\_hw.h, 662
- atcacert\_host\_sw.c, 662
- atcacert\_host\_sw.h, 663
- atcacert\_is\_device\_loc\_overlap
  - Certificate manipulation methods (atcacert\_), 187
- ATCACERT\_MAX
  - atcacert\_def.c, 655
- atcacert\_max\_cert\_size
  - Certificate manipulation methods (atcacert\_), 187
- atcacert\_merge\_device\_loc
  - Certificate manipulation methods (atcacert\_), 188
- ATCACERT\_MIN
  - atcacert\_def.c, 656
- atcacert\_pem.c, 664
  - atcacert\_decode\_pem, 664
  - atcacert\_decode\_pem\_cert, 665
  - atcacert\_decode\_pem\_csr, 665
  - atcacert\_encode\_pem, 666
  - atcacert\_encode\_pem\_cert, 666
  - atcacert\_encode\_pem\_csr, 667
- atcacert\_pem.h, 667
  - atcacert\_decode\_pem, 669
  - atcacert\_decode\_pem\_cert, 669
  - atcacert\_decode\_pem\_csr, 670
  - atcacert\_encode\_pem, 670
  - atcacert\_encode\_pem\_cert, 671
  - atcacert\_encode\_pem\_csr, 671
  - PEM\_CERT\_BEGIN, 668
  - PEM\_CERT\_END, 668
  - PEM\_CSR\_BEGIN, 668
  - PEM\_CSR\_END, 669
- atcacert\_public\_key\_add\_padding
  - Certificate manipulation methods (atcacert\_), 188
- atcacert\_public\_key\_remove\_padding
  - Certificate manipulation methods (atcacert\_), 189
- atcacert\_read\_cert

- Certificate manipulation methods (atcacert\_), 189
- atcacert\_read\_cert\_size
  - Certificate manipulation methods (atcacert\_), 191
- atcacert\_read\_device\_loc
  - Certificate manipulation methods (atcacert\_), 191
- atcacert\_read\_subj\_key\_id
  - Certificate manipulation methods (atcacert\_), 192
- atcacert\_set\_auth\_key\_id
  - Certificate manipulation methods (atcacert\_), 192
- atcacert\_set\_auth\_key\_id\_raw
  - Certificate manipulation methods (atcacert\_), 192
- atcacert\_set\_cert\_element
  - Certificate manipulation methods (atcacert\_), 193
- atcacert\_set\_cert\_sn
  - Certificate manipulation methods (atcacert\_), 193
- atcacert\_set\_comp\_cert
  - Certificate manipulation methods (atcacert\_), 194
- atcacert\_set\_expire\_date
  - Certificate manipulation methods (atcacert\_), 195
- atcacert\_set\_issue\_date
  - Certificate manipulation methods (atcacert\_), 195
- atcacert\_set\_signature
  - Certificate manipulation methods (atcacert\_), 196
- atcacert\_set\_signer\_id
  - Certificate manipulation methods (atcacert\_), 196
- atcacert\_set\_subj\_public\_key
  - Certificate manipulation methods (atcacert\_), 197
- atcacert\_std\_cert\_element\_e
  - Certificate manipulation methods (atcacert\_), 166
- atcacert\_std\_cert\_element\_t
  - Certificate manipulation methods (atcacert\_), 163
- atcacert\_tm\_utc\_s, 454
  - tm\_hour, 454
  - tm\_mday, 454
  - tm\_min, 454
  - tm\_mon, 455
  - tm\_sec, 455
  - tm\_year, 455
- atcacert\_tm\_utc\_t
  - Certificate manipulation methods (atcacert\_), 163
- atcacert\_transform\_data
  - Certificate manipulation methods (atcacert\_), 197
- atcacert\_transform\_e
  - Certificate manipulation methods (atcacert\_), 166
- atcacert\_transform\_t
  - Certificate manipulation methods (atcacert\_), 163
- atcacert\_verify\_cert\_hw
  - Certificate manipulation methods (atcacert\_), 198
- atcacert\_verify\_cert\_sw
  - Certificate manipulation methods (atcacert\_), 198
- atcacert\_verify\_response\_hw
  - Certificate manipulation methods (atcacert\_), 199
- atcacert\_verify\_response\_sw
  - Certificate manipulation methods (atcacert\_), 199
- atcacert\_write\_cert
  - Certificate manipulation methods (atcacert\_), 200
- ATCACCommand
  - ATCACCommand (atca\_), 122
- ATCACCommand (atca\_), 122
  - ATCACCommand, 122
  - deleteATCACCommand, 122
  - initATCACCommand, 123
  - newATCACCommand, 123
- atcacustom
  - ATCAIfaceCfg, 460
- ATCADevice
  - ATCADevice (atca\_), 142
- ATCADevice (atca\_), 124
  - ATCA\_AES\_ENABLE\_EN\_MASK, 127
  - ATCA\_AES\_ENABLE\_EN\_SHIFT, 127
  - ATCA\_CHIP\_MODE\_CLK\_DIV, 127
  - ATCA\_CHIP\_MODE\_CLK\_DIV\_MASK, 127
  - ATCA\_CHIP\_MODE\_CLK\_DIV\_SHIFT, 127
  - ATCA\_CHIP\_MODE\_I2C\_EXTRA\_MASK, 127
  - ATCA\_CHIP\_MODE\_I2C\_EXTRA\_SHIFT, 127
  - ATCA\_CHIP\_MODE\_TTL\_EN\_MASK, 128
  - ATCA\_CHIP\_MODE\_TTL\_EN\_SHIFT, 128
  - ATCA\_CHIP\_MODE\_WDG\_LONG\_MASK, 128
  - ATCA\_CHIP\_MODE\_WDG\_LONG\_SHIFT, 128
  - ATCA\_CHIP\_OPT\_ECDH\_PROT, 128
  - ATCA\_CHIP\_OPT\_ECDH\_PROT\_MASK, 128
  - ATCA\_CHIP\_OPT\_ECDH\_PROT\_SHIFT, 128
  - ATCA\_CHIP\_OPT\_IO\_PROT\_EN\_MASK, 129
  - ATCA\_CHIP\_OPT\_IO\_PROT\_EN\_SHIFT, 129
  - ATCA\_CHIP\_OPT\_IO\_PROT\_KEY, 129
  - ATCA\_CHIP\_OPT\_IO\_PROT\_KEY\_MASK, 129
  - ATCA\_CHIP\_OPT\_IO\_PROT\_KEY\_SHIFT, 129
  - ATCA\_CHIP\_OPT\_KDF\_AES\_EN\_MASK, 129
  - ATCA\_CHIP\_OPT\_KDF\_AES\_EN\_SHIFT, 129
  - ATCA\_CHIP\_OPT\_KDF\_PROT, 130
  - ATCA\_CHIP\_OPT\_KDF\_PROT\_MASK, 130
  - ATCA\_CHIP\_OPT\_KDF\_PROT\_SHIFT, 130
  - ATCA\_CHIP\_OPT\_POST\_EN\_MASK, 130
  - ATCA\_CHIP\_OPT\_POST\_EN\_SHIFT, 130
  - ATCA\_COUNTER\_MATCH\_EN\_MASK, 130
  - ATCA\_COUNTER\_MATCH\_EN\_SHIFT, 130
  - ATCA\_COUNTER\_MATCH\_KEY, 131
  - ATCA\_COUNTER\_MATCH\_KEY\_MASK, 131
  - ATCA\_COUNTER\_MATCH\_KEY\_SHIFT, 131
  - ATCA\_DEV\_UNKNOWN, 143
  - ATCA\_I2C\_ENABLE\_EN\_MASK, 131
  - ATCA\_I2C\_ENABLE\_EN\_SHIFT, 131
  - ATCA\_KEY\_CONFIG\_AUTH\_KEY, 131
  - ATCA\_KEY\_CONFIG\_AUTH\_KEY\_MASK, 131
  - ATCA\_KEY\_CONFIG\_AUTH\_KEY\_SHIFT, 132
  - ATCA\_KEY\_CONFIG\_KEY\_TYPE, 132
  - ATCA\_KEY\_CONFIG\_KEY\_TYPE\_MASK, 132
  - ATCA\_KEY\_CONFIG\_KEY\_TYPE\_SHIFT, 132
  - ATCA\_KEY\_CONFIG\_LOCKABLE\_MASK, 132
  - ATCA\_KEY\_CONFIG\_LOCKABLE\_SHIFT, 132
  - ATCA\_KEY\_CONFIG\_PERSIST\_DISABLE\_MASK, 132
  - ATCA\_KEY\_CONFIG\_PERSIST\_DISABLE\_SHIFT, 133
  - ATCA\_KEY\_CONFIG\_PRIVATE\_MASK, 133
  - ATCA\_KEY\_CONFIG\_PRIVATE\_SHIFT, 133

- ATCA\_KEY\_CONFIG\_PUB\_INFO\_MASK, [133](#)
- ATCA\_KEY\_CONFIG\_PUB\_INFO\_SHIFT, [133](#)
- ATCA\_KEY\_CONFIG\_REQ\_AUTH\_MASK, [133](#)
- ATCA\_KEY\_CONFIG\_REQ\_AUTH\_SHIFT, [133](#)
- ATCA\_KEY\_CONFIG\_REQ\_RANDOM\_MASK, [133](#)
- ATCA\_KEY\_CONFIG\_REQ\_RANDOM\_SHIFT, [134](#)
- ATCA\_KEY\_CONFIG\_RFU\_MASK, [134](#)
- ATCA\_KEY\_CONFIG\_RFU\_SHIFT, [134](#)
- ATCA\_KEY\_CONFIG\_X509\_ID, [134](#)
- ATCA\_KEY\_CONFIG\_X509\_ID\_MASK, [134](#)
- ATCA\_KEY\_CONFIG\_X509\_ID\_SHIFT, [134](#)
- ATCA\_PACKED, [134](#)
- ATCA\_SECURE\_BOOT\_DIGEST, [135](#)
- ATCA\_SECURE\_BOOT\_DIGEST\_MASK, [135](#)
- ATCA\_SECURE\_BOOT\_DIGEST\_SHIFT, [135](#)
- ATCA\_SECURE\_BOOT\_MODE, [135](#)
- ATCA\_SECURE\_BOOT\_MODE\_MASK, [135](#)
- ATCA\_SECURE\_BOOT\_MODE\_SHIFT, [135](#)
- ATCA\_SECURE\_BOOT\_PERSIST\_EN\_MASK, [135](#)
- ATCA\_SECURE\_BOOT\_PERSIST\_EN\_SHIFT, [136](#)
- ATCA\_SECURE\_BOOT\_PUB\_KEY, [136](#)
- ATCA\_SECURE\_BOOT\_PUB\_KEY\_MASK, [136](#)
- ATCA\_SECURE\_BOOT\_PUB\_KEY\_SHIFT, [136](#)
- ATCA\_SECURE\_BOOT\_RAND\_NONCE\_MASK, [136](#)
- ATCA\_SECURE\_BOOT\_RAND\_NONCE\_SHIFT, [136](#)
- ATCA\_SLOT\_CONFIG\_ECDH\_MASK, [136](#)
- ATCA\_SLOT\_CONFIG\_ECDH\_SHIFT, [137](#)
- ATCA\_SLOT\_CONFIG\_ENCRYPTED\_READ\_MASK, [137](#)
- ATCA\_SLOT\_CONFIG\_ENCRYPTED\_READ\_SHIFT, [137](#)
- ATCA\_SLOT\_CONFIG\_EXT\_SIG\_MASK, [137](#)
- ATCA\_SLOT\_CONFIG\_EXT\_SIG\_SHIFT, [137](#)
- ATCA\_SLOT\_CONFIG\_GEN\_KEY\_MASK, [137](#)
- ATCA\_SLOT\_CONFIG\_GEN\_KEY\_SHIFT, [137](#)
- ATCA\_SLOT\_CONFIG\_INT\_SIG\_MASK, [137](#)
- ATCA\_SLOT\_CONFIG\_INT\_SIG\_SHIFT, [138](#)
- ATCA\_SLOT\_CONFIG\_IS\_SECRET\_MASK, [138](#)
- ATCA\_SLOT\_CONFIG\_IS\_SECRET\_SHIFT, [138](#)
- ATCA\_SLOT\_CONFIG\_LIMITED\_USE\_MASK, [138](#)
- ATCA\_SLOT\_CONFIG\_LIMITED\_USE\_SHIFT, [138](#)
- ATCA\_SLOT\_CONFIG\_NOMAC\_MASK, [138](#)
- ATCA\_SLOT\_CONFIG\_NOMAC\_SHIFT, [138](#)
- ATCA\_SLOT\_CONFIG\_PRIV\_WRITE\_MASK, [138](#)
- ATCA\_SLOT\_CONFIG\_PRIV\_WRITE\_SHIFT, [139](#)
- ATCA\_SLOT\_CONFIG\_READKEY, [139](#)
- ATCA\_SLOT\_CONFIG\_READKEY\_MASK, [139](#)
- ATCA\_SLOT\_CONFIG\_READKEY\_SHIFT, [139](#)
- ATCA\_SLOT\_CONFIG\_WRITE\_CONFIG, [139](#)
- ATCA\_SLOT\_CONFIG\_WRITE\_CONFIG\_MASK, [139](#)
- ATCA\_SLOT\_CONFIG\_WRITE\_CONFIG\_SHIFT, [139](#)
- ATCA\_SLOT\_CONFIG\_WRITE\_ECDH\_MASK, [140](#)
- ATCA\_SLOT\_CONFIG\_WRITE\_ECDH\_SHIFT, [140](#)
- ATCA\_SLOT\_CONFIG\_WRITE\_KEY, [140](#)
- ATCA\_SLOT\_CONFIG\_WRITE\_KEY\_MASK, [140](#)
- ATCA\_SLOT\_CONFIG\_WRITE\_KEY\_SHIFT, [140](#)
- ATCA\_SLOT\_LOCKED, [140](#)
- ATCA\_USE\_LOCK\_ENABLE\_MASK, [140](#)
- ATCA\_USE\_LOCK\_ENABLE\_SHIFT, [141](#)
- ATCA\_USE\_LOCK\_KEY\_MASK, [141](#)
- ATCA\_USE\_LOCK\_KEY\_SHIFT, [141](#)
- ATCA\_VOL\_KEY\_PERM\_EN\_MASK, [141](#)
- ATCA\_VOL\_KEY\_PERM\_EN\_SHIFT, [141](#)
- ATCA\_VOL\_KEY\_PERM\_SLOT, [141](#)
- ATCA\_VOL\_KEY\_PERM\_SLOT\_MASK, [141](#)
- ATCA\_VOL\_KEY\_PERM\_SLOT\_SHIFT, [142](#)
- ATCADevice, [142](#)
- ATCADeviceType, [142](#)
- ATECC108A, [143](#)
- ATECC508A, [143](#)
- atecc508a\_config\_t, [142](#)
- ATECC608, [143](#)
- atecc608\_config\_t, [142](#)
- ATECC608A, [143](#)
- ATECC608B, [143](#)
- atGetCommands, [143](#)
- atGetIFace, [143](#)
- ATSHA204A, [143](#)
- atsha204a\_config\_t, [142](#)
- ATSHA206A, [143](#)
- deleteATCADevice, [144](#)
- initATCADevice, [144](#)
- newATCADevice, [144](#)
- releaseATCADevice, [145](#)
- TA100, [143](#)
- ATCADeviceType
  - ATCADevice (atca\_), [142](#)
- atcah\_check\_mac
  - Host side crypto methods (atcah\_), [317](#)
- atcah\_config\_to\_sign\_internal
  - Host side crypto methods (atcah\_), [317](#)
- atcah\_decrypt
  - Host side crypto methods (atcah\_), [318](#)
- atcah\_derive\_key
  - Host side crypto methods (atcah\_), [318](#)
- atcah\_derive\_key\_mac
  - Host side crypto methods (atcah\_), [319](#)
- atcah\_encode\_counter\_match
  - Host side crypto methods (atcah\_), [319](#)
- atcah\_gen\_dig
  - Host side crypto methods (atcah\_), [320](#)
- atcah\_gen\_key\_msg
  - Host side crypto methods (atcah\_), [320](#)
- atcah\_gen\_mac

---

- Host side crypto methods (atcah\_), 321
- atcah\_hmac
  - Host side crypto methods (atcah\_), 321
- atcah\_include\_data
  - Host side crypto methods (atcah\_), 321
- atcah\_io\_decrypt
  - Host side crypto methods (atcah\_), 322
- atcah\_mac
  - Host side crypto methods (atcah\_), 322
- atcah\_nonce
  - Host side crypto methods (atcah\_), 323
- atcah\_privwrite\_auth\_mac
  - Host side crypto methods (atcah\_), 323
- atcah\_secureboot\_enc
  - Host side crypto methods (atcah\_), 323
- atcah\_secureboot\_mac
  - Host side crypto methods (atcah\_), 324
- atcah\_sha256
  - Host side crypto methods (atcah\_), 324
- atcah\_sign\_internal\_msg
  - Host side crypto methods (atcah\_), 324
- atcah\_verify\_mac
  - Host side crypto methods (atcah\_), 325
- atcah\_write\_auth\_mac
  - Host side crypto methods (atcah\_), 325
- ATCAHAL\_t, 455
  - hal\_data, 456
  - halidle, 456
  - halinit, 456
  - halpostinit, 456
  - halreceive, 456
  - halrelease, 456
  - halsend, 456
  - halsleep, 456
  - halwake, 457
- atcahid, 457
  - ATCAIfaceCfg, 460
  - kits, 457
  - num\_kits\_found, 457
- atcahid\_t
  - Hardware abstraction layer (hal\_), 267
- atcai2c
  - ATCAIfaceCfg, 460
- atcal2Cmaster, 458
  - bus\_index, 458
  - i2c\_file, 458
  - id, 458
  - ref\_ct, 458
  - twi\_id, 458
  - twi\_master\_instance, 459
- ATCAI2CMaster\_t
  - hal\_esp32\_i2c.c, 837
  - Hardware abstraction layer (hal\_), 268
- ATCAIface
  - ATCAIface (atca\_), 147
- ATCAIface (atca\_), 146
  - ATCA\_CUSTOM\_IFACE, 147
  - ATCA\_HID\_IFACE, 147
  - ATCA\_I2C\_IFACE, 147
  - ATCA\_KIT\_AUTO\_IFACE, 147
  - ATCA\_KIT\_I2C\_IFACE, 147
  - ATCA\_KIT\_SPI\_IFACE, 147
  - ATCA\_KIT\_SWI\_IFACE, 147
  - ATCA\_KIT\_UNKNOWN\_IFACE, 147
  - ATCA\_SPI\_IFACE, 147
  - ATCA\_SWI\_IFACE, 147
  - ATCA\_UART\_IFACE, 147
  - ATCA\_UNKNOWN\_IFACE, 147
  - ATCAIface, 147
  - ATCAIfaceType, 147
  - ATCAKitType, 147
  - atgetifacecfg, 148
  - atgetifacehaldat, 148
  - atidle, 148
  - atinit, 149
  - atpostinit, 149
  - atreceive, 149
  - atsend, 150
  - atsleep, 150
  - atwake, 150
  - deleteATCAIface, 151
  - initATCAIface, 151
  - newATCAIface, 151
  - releaseATCAIface, 152
- ATCAIfaceCfg, 459
  - atcacustom, 460
  - atcahid, 460
  - atcai2c, 460
  - atcaspi, 460
  - atcaswi, 460
  - atcauart, 461
  - baud, 461
  - bus, 461
  - cfg\_data, 461
  - dev\_identity, 461
  - dev\_interface, 461
  - devtype, 461
  - halidle, 461
  - halinit, 462
  - halpostinit, 462
  - halreceive, 462
  - halrelease, 462
  - halsend, 462
  - halsleep, 462
  - halwake, 462
  - idx, 462
  - iface\_type, 463
  - packetsize, 463
  - parity, 463
  - pid, 463
  - port, 463
  - rx\_retries, 463
  - select\_pin, 463
  - slave\_address, 463
  - stopbits, 464
  - vid, 464

- wake\_delay, [464](#)
  - wordsize, [464](#)
- ATCAIfaceType
  - ATCAIface (atca\_), [147](#)
- ATCAKitType
  - ATCAIface (atca\_), [147](#)
- atCalcCrc
  - calib\_command.c, [687](#)
  - calib\_command.h, [789](#)
- ATCAPacket, [464](#)
  - \_reserved, [464](#)
  - data, [465](#)
  - execTime, [465](#)
  - opcode, [465](#)
  - param1, [465](#)
  - param2, [465](#)
  - txsize, [465](#)
- atcaspi
  - ATCAIfaceCfg, [460](#)
- atcaSPImaster, [465](#)
  - ref\_ct, [466](#)
  - spi\_file, [466](#)
- ATCASPIMaster\_t
  - Hardware abstraction layer (hal\_), [268](#)
- atcaswi
  - ATCAIfaceCfg, [460](#)
- atcaSWImaster, [466](#)
  - bus\_index, [466](#)
  - ref\_ct, [467](#)
  - sercom\_core\_freq, [467](#)
  - usart\_instance, [467](#)
  - USART\_SWI, [467](#)
- ATCASWIMaster\_t
  - Hardware abstraction layer (hal\_), [268](#)
- atcauart
  - ATCAIfaceCfg, [461](#)
- atCheckCrc
  - calib\_command.c, [688](#)
  - calib\_command.h, [789](#)
- atCheckMAC
  - calib\_command.c, [688](#)
  - calib\_command.h, [789](#)
- atCounter
  - calib\_command.c, [688](#)
  - calib\_command.h, [790](#)
- atCRC
  - calib\_command.c, [689](#)
  - calib\_command.h, [790](#)
- atDeriveKey
  - calib\_command.c, [689](#)
  - calib\_command.h, [790](#)
- ATECC108A
  - ATCADevice (atca\_), [143](#)
- ATECC508A
  - ATCADevice (atca\_), [143](#)
- atecc508a\_config\_t
  - ATCADevice (atca\_), [142](#)
- ATECC608
  - ATCADevice (atca\_), [143](#)
- ATECC608A
  - ATCADevice (atca\_), [143](#)
- ATECC608B
  - ATCADevice (atca\_), [143](#)
- atECDH
  - calib\_command.c, [690](#)
  - calib\_command.h, [791](#)
- atGenDig
  - calib\_command.c, [690](#)
  - calib\_command.h, [791](#)
- atGenKey
  - calib\_command.c, [690](#)
  - calib\_command.h, [792](#)
- atGetCommands
  - ATCADevice (atca\_), [143](#)
- atGetIFace
  - ATCADevice (atca\_), [143](#)
- atgetifacecfg
  - ATCAIface (atca\_), [148](#)
- atgetifacehaldat
  - ATCAIface (atca\_), [148](#)
- atHMAC
  - calib\_command.c, [691](#)
  - calib\_command.h, [792](#)
- atidle
  - atca\_iface, [426](#)
  - ATCAIface (atca\_), [148](#)
- atInfo
  - calib\_command.c, [691](#)
  - calib\_command.h, [792](#)
- atinit
  - atca\_iface, [427](#)
  - ATCAIface (atca\_), [149](#)
- atIsECCFamily
  - calib\_command.c, [691](#)
  - calib\_command.h, [793](#)
- atIsSHAFamily
  - calib\_command.c, [692](#)
  - calib\_command.h, [793](#)
- atKDF
  - calib\_command.c, [692](#)
  - calib\_command.h, [794](#)
- atLock
  - calib\_command.c, [693](#)
  - calib\_command.h, [794](#)
- atMAC
  - calib\_command.c, [693](#)
  - calib\_command.h, [794](#)
- atNonce
  - calib\_command.c, [693](#)
  - calib\_command.h, [795](#)
- atPause
  - calib\_command.c, [694](#)



- calib\_command.h, 795
- atpostinit
  - atca\_iface, 427
  - ATCAIface (atca\_), 149
- atPrivWrite
  - calib\_command.c, 694
  - calib\_command.h, 795
- atRandom
  - calib\_command.c, 694
  - calib\_command.h, 796
- atRead
  - calib\_command.c, 695
  - calib\_command.h, 796
- atreceive
  - atca\_iface, 427
  - ATCAIface (atca\_), 149
- atSecureBoot
  - calib\_command.c, 695
  - calib\_command.h, 797
- atSelfTest
  - calib\_command.c, 696
  - calib\_command.h, 797
- atsend
  - atca\_iface, 427
  - ATCAIface (atca\_), 150
- atSHA
  - calib\_command.c, 696
  - calib\_command.h, 797
- ATSHA204A
  - ATCADevice (atca\_), 143
- atsha204a\_config\_t
  - ATCADevice (atca\_), 142
- ATSHA206A
  - ATCADevice (atca\_), 143
- atSign
  - calib\_command.c, 696
  - calib\_command.h, 799
- atsleep
  - atca\_iface, 427
  - ATCAIface (atca\_), 150
- attrib\_count
  - \_pkcs11\_session\_ctx, 404
- attrib\_f
  - pkcs11\_attrib.h, 897
- attrib\_list
  - \_pkcs11\_session\_ctx, 404
- attributes
  - \_pkcs11\_object, 401
- Attributes (pkcs11\_attrib\_), 335
  - C\_CancelFunction, 343
  - C\_CloseAllSessions, 343
  - C\_CloseSession, 343
  - C\_CopyObject, 343
  - C\_CreateObject, 343
  - C\_Decrypt, 344
  - C\_DecryptDigestUpdate, 344
  - C\_DecryptFinal, 344
  - C\_DecryptInit, 344
  - C\_DecryptUpdate, 345
  - C\_DecryptVerifyUpdate, 345
  - C\_DeriveKey, 345
  - C\_DestroyObject, 345
  - C\_Digest, 346
  - C\_DigestEncryptUpdate, 346
  - C\_DigestFinal, 346
  - C\_DigestInit, 346
  - C\_DigestKey, 347
  - C\_DigestUpdate, 347
  - C\_Encrypt, 347
  - C\_EncryptFinal, 347
  - C\_EncryptInit, 347
  - C\_EncryptUpdate, 348
  - C\_Finalize, 348
  - C\_FindObjects, 348
  - C\_FindObjectsFinal, 348
  - C\_FindObjectsInit, 348
  - C\_GenerateKey, 349
  - C\_GenerateKeyPair, 349
  - C\_GenerateRandom, 349
  - C\_GetAttributeValue, 349
  - C\_GetFunctionList, 350
  - C\_GetFunctionStatus, 350
  - C\_GetInfo, 350
  - C\_GetMechanismInfo, 350
  - C\_GetMechanismList, 350
  - C\_GetObjectSize, 351
  - C\_GetOperationState, 351
  - C\_GetSessionInfo, 351
  - C\_GetSlotInfo, 351
  - C\_GetSlotList, 351
  - C\_GetTokenInfo, 352
  - C\_Initialize, 352
  - C\_InitPIN, 352
  - C\_InitToken, 352
  - C\_Login, 352
  - C\_Logout, 353
  - C\_OpenSession, 353
  - C\_SeedRandom, 353
  - C\_SetAttributeValue, 353
  - C\_SetOperationState, 354
  - C\_SetPIN, 354
  - C\_Sign, 354
  - C\_SignEncryptUpdate, 354
  - C\_SignFinal, 355
  - C\_SignInit, 355
  - C\_SignRecover, 355
  - C\_SignRecoverInit, 355
  - C\_SignUpdate, 356
  - C\_UnwrapKey, 356
  - C\_Verify, 356
  - C\_VerifyFinal, 356
  - C\_VerifyInit, 357
  - C\_VerifyRecover, 357
  - C\_VerifyRecoverInit, 357
  - C\_VerifyUpdate, 357
  - C\_WaitForSlotEvent, 357

- C\_WrapKey, 358
- PKCS11\_MECH\_ECC508\_EC\_CAPABILITY, 342
- pkcs11\_mech\_table\_e, 342
- pkcs11\_mech\_table\_ptr, 342
- pkcs11\_attr\_empty, 358
- pkcs11\_attr\_false, 358
- pkcs11\_attr\_fill, 358
- pkcs11\_attr\_true, 358
- pkcs11\_attr\_value, 359
- pkcs11\_cert\_get\_authority\_key\_id, 359
- pkcs11\_cert\_get\_encoded, 359
- pkcs11\_cert\_get\_subject, 359
- pkcs11\_cert\_get\_subject\_key\_id, 359
- pkcs11\_cert\_get\_trusted\_flag, 359
- pkcs11\_cert\_get\_type, 360
- pkcs11\_cert\_wtlspublic\_attributes, 374
- pkcs11\_cert\_wtlspublic\_attributes\_count, 374
- pkcs11\_cert\_x509\_attributes, 374
- pkcs11\_cert\_x509\_attributes\_count, 374
- pkcs11\_cert\_x509\_write, 360
- pkcs11\_cert\_x509public\_attributes, 374
- pkcs11\_cert\_x509public\_attributes\_count, 374
- pkcs11\_config\_cert, 360
- pkcs11\_config\_init\_cert, 360
- pkcs11\_config\_init\_private, 360
- pkcs11\_config\_init\_public, 360
- pkcs11\_config\_key, 361
- pkcs11\_config\_load, 361
- pkcs11\_config\_load\_objects, 361
- pkcs11\_config\_remove\_object, 361
- pkcs11\_deinit, 361
- pkcs11\_find\_continue, 361
- pkcs11\_find\_finish, 362
- pkcs11\_find\_get\_attribute, 362
- pkcs11\_find\_init, 362
- pkcs11\_get\_context, 362
- pkcs11\_get\_lib\_info, 362
- pkcs11\_get\_session\_context, 362
- pkcs11\_init, 363
- pkcs11\_init\_check, 363
- pkcs11\_key\_derive, 363
- pkcs11\_key\_ec\_private\_attributes, 374
- pkcs11\_key\_ec\_public\_attributes, 375
- pkcs11\_key\_generate\_pair, 363
- pkcs11\_key\_private\_attributes, 375
- pkcs11\_key\_private\_attributes\_count, 375
- pkcs11\_key\_public\_attributes, 375
- pkcs11\_key\_public\_attributes\_count, 375
- pkcs11\_key\_rsa\_private\_attributes, 375
- pkcs11\_key\_secret\_attributes, 376
- pkcs11\_key\_secret\_attributes\_count, 376
- pkcs11\_key\_write, 364
- pkcs11\_lib\_description, 376
- pkcs11\_lib\_manufacturer\_id, 376
- pkcs11\_lock\_context, 364
- pkcs11\_mech\_get\_list, 364
- pkcs11\_object\_alloc, 364
- pkcs11\_object\_cache, 376
- pkcs11\_object\_check, 364
- pkcs11\_object\_create, 365
- pkcs11\_object\_deinit, 365
- pkcs11\_object\_destroy, 365
- pkcs11\_object\_find, 365
- pkcs11\_object\_free, 365
- pkcs11\_object\_get\_class, 365
- pkcs11\_object\_get\_destroyable, 366
- pkcs11\_object\_get\_handle, 366
- pkcs11\_object\_get\_name, 366
- pkcs11\_object\_get\_size, 366
- pkcs11\_object\_get\_type, 366
- pkcs11\_object\_load\_handle\_info, 366
- pkcs11\_object\_monotonic\_attributes, 376
- pkcs11\_object\_monotonic\_attributes\_count, 377
- pkcs11\_os\_create\_mutex, 367
- pkcs11\_os\_destroy\_mutex, 367
- pkcs11\_os\_lock\_mutex, 367
- pkcs11\_os\_unlock\_mutex, 367
- pkcs11\_session\_check, 367
- pkcs11\_session\_close, 367
- pkcs11\_session\_closeall, 368
- pkcs11\_session\_get\_info, 368
- pkcs11\_session\_login, 368
- pkcs11\_session\_logout, 368
- pkcs11\_session\_open, 368
- pkcs11\_signature\_sign, 368
- pkcs11\_signature\_sign\_continue, 369
- pkcs11\_signature\_sign\_finish, 369
- pkcs11\_signature\_sign\_init, 369
- pkcs11\_signature\_verify, 369
- pkcs11\_signature\_verify\_continue, 370
- pkcs11\_signature\_verify\_finish, 370
- pkcs11\_signature\_verify\_init, 370
- pkcs11\_slot\_config, 370
- pkcs11\_slot\_get\_context, 370
- pkcs11\_slot\_get\_info, 371
- pkcs11\_slot\_get\_list, 371
- pkcs11\_slot\_init, 371
- pkcs11\_slot\_initslots, 371
- pkcs11\_token\_convert\_pin\_to\_key, 371
- pkcs11\_token\_get\_access\_type, 371
- pkcs11\_token\_get\_info, 372
- pkcs11\_token\_get\_storage, 372
- pkcs11\_token\_get\_writable, 372
- pkcs11\_token\_init, 372
- pkcs11\_token\_random, 372
- pkcs11\_token\_set\_pin, 372
- pkcs11\_unlock\_context, 373
- pkcs11\_util\_convert\_rv, 373
- pkcs11\_util\_escape\_string, 373
- pkcs11\_util\_memset, 373
- pkcs11\_util\_memset, 373
- pkcs\_mech\_get\_info, 373
- TABLE\_SIZE, 342
- atUpdateExtra
  - calib\_command.c, 697
  - calib\_command.h, 799
- atVerify



- calib\_command.c, 697
- calib\_command.h, 799
- atwake
  - atca\_iface, 427
  - ATCAIface (atca\_), 150
- atWrite
  - calib\_command.c, 698
  - calib\_command.h, 800
- auth\_mac
  - atca\_write\_mac\_in\_out, 444
- B64\_IS\_EQUAL
  - atca\_helpers.c, 599
- B64\_IS\_INVALID
  - atca\_helpers.c, 599
- base64Char
  - atca\_helpers.c, 603
  - atca\_helpers.h, 615
- base64Index
  - atca\_helpers.c, 605
  - atca\_helpers.h, 615
- Basic Crypto API methods (atcab\_), 51
  - \_atcab\_exit, 60
  - \_gDevice, 120
  - atca\_aes\_gcm\_ctx\_t, 59
  - ATCA\_AES\_GCM\_IV\_STD\_LENGTH, 59
  - atca\_basic\_aes\_gcm\_version, 120
  - atca\_execute\_command, 59
  - atcab\_aes, 60
  - atcab\_aes\_cbc\_decrypt\_block, 60
  - atcab\_aes\_cbc\_encrypt\_block, 61
  - atcab\_aes\_cbc\_encrypt\_block\_ext, 61
  - atcab\_aes\_cbc\_init, 61
  - atcab\_aes\_cbc\_init\_ext, 62
  - atcab\_aes\_cmac\_finish, 62
  - atcab\_aes\_cmac\_init, 63
  - atcab\_aes\_cmac\_init\_ext, 63
  - atcab\_aes\_cmac\_update, 64
  - atcab\_aes\_ctr\_block, 64
  - atcab\_aes\_ctr\_decrypt\_block, 64
  - atcab\_aes\_ctr\_encrypt\_block, 65
  - atcab\_aes\_ctr\_increment, 65
  - atcab\_aes\_ctr\_init, 66
  - atcab\_aes\_ctr\_init\_ext, 66
  - atcab\_aes\_ctr\_init\_rand, 67
  - atcab\_aes\_ctr\_init\_rand\_ext, 67
  - atcab\_aes\_decrypt, 68
  - atcab\_aes\_decrypt\_ext, 68
  - atcab\_aes\_encrypt, 69
  - atcab\_aes\_encrypt\_ext, 69
  - atcab\_aes\_gcm\_aad\_update, 70
  - atcab\_aes\_gcm\_decrypt\_finish, 70
  - atcab\_aes\_gcm\_decrypt\_update, 71
  - atcab\_aes\_gcm\_encrypt\_finish, 71
  - atcab\_aes\_gcm\_encrypt\_update, 72
  - atcab\_aes\_gcm\_init, 72
  - atcab\_aes\_gcm\_init\_rand, 73
  - atcab\_aes\_gfm, 73
  - atcab\_cfg\_discover, 59
  - atcab\_challenge, 74
  - atcab\_challenge\_seed\_update, 74
  - atcab\_checkmac, 75
  - atcab\_cmp\_config\_zone, 75
  - atcab\_counter, 76
  - atcab\_counter\_increment, 76
  - atcab\_counter\_read, 76
  - atcab\_derivekey, 77
  - atcab\_ecdh, 77
  - atcab\_ecdh\_base, 78
  - atcab\_ecdh\_enc, 78
  - atcab\_ecdh\_ioenc, 79
  - atcab\_ecdh\_tempkey, 79
  - atcab\_ecdh\_tempkey\_ioenc, 80
  - atcab\_gendig, 80
  - atcab\_genkey, 81
  - atcab\_genkey\_base, 81
  - atcab\_get\_addr, 59
  - atcab\_get\_device, 82
  - atcab\_get\_device\_type, 82
  - atcab\_get\_device\_type\_ext, 82
  - atcab\_get\_pubkey, 82
  - atcab\_get\_zone\_size, 83
  - atcab\_hmac, 83
  - atcab\_hw\_sha2\_256, 84
  - atcab\_hw\_sha2\_256\_finish, 84
  - atcab\_hw\_sha2\_256\_init, 84
  - atcab\_hw\_sha2\_256\_update, 85
  - atcab\_idle, 85
  - atcab\_info, 85
  - atcab\_info\_base, 86
  - atcab\_info\_get\_latch, 86
  - atcab\_info\_set\_latch, 86
  - atcab\_init, 87
  - atcab\_init\_device, 87
  - atcab\_init\_ext, 88
  - atcab\_is\_ca\_device, 88
  - atcab\_is\_config\_locked, 88
  - atcab\_is\_data\_locked, 89
  - atcab\_is\_locked, 89
  - atcab\_is\_slot\_locked, 89
  - atcab\_is\_ta\_device, 90
  - atcab\_kdf, 90
  - atcab\_lock, 91
  - atcab\_lock\_config\_zone, 91
  - atcab\_lock\_config\_zone\_crc, 91
  - atcab\_lock\_data\_slot, 92
  - atcab\_lock\_data\_zone, 92
  - atcab\_lock\_data\_zone\_crc, 92
  - atcab\_mac, 93
  - atcab\_nonce, 93
  - atcab\_nonce\_base, 94
  - atcab\_nonce\_load, 94
  - atcab\_nonce\_rand, 95
  - atcab\_printbin, 95
  - atcab\_priv\_write, 95
  - atcab\_random, 96
  - atcab\_random\_ext, 96

- atcab\_read\_bytes\_zone, 96
- atcab\_read\_config\_zone, 97
- atcab\_read\_enc, 97
- atcab\_read\_pubkey, 98
- atcab\_read\_serial\_number, 98
- atcab\_read\_sig, 99
- atcab\_read\_zone, 99
- atcab\_release, 99
- atcab\_release\_ext, 100
- atcab\_secureboot, 100
- atcab\_secureboot\_mac, 101
- atcab\_selftest, 101
- atcab\_sha, 102
- atcab\_sha\_base, 102
- atcab\_sha\_end, 103
- atcab\_sha\_hmac, 103
- atcab\_sha\_hmac\_finish, 104
- atcab\_sha\_hmac\_init, 104
- atcab\_sha\_hmac\_update, 104
- atcab\_sha\_read\_context, 105
- atcab\_sha\_start, 105
- atcab\_sha\_update, 105
- atcab\_sha\_write\_context, 106
- atcab\_sign, 106
- atcab\_sign\_base, 107
- atcab\_sign\_internal, 107
- atcab\_sleep, 108
- atcab\_updateextra, 108
- atcab\_verify, 108
- atcab\_verify\_extern, 109
- atcab\_verify\_extern\_mac, 110
- atcab\_verify\_invalidate, 110
- atcab\_verify\_stored, 111
- atcab\_verify\_stored\_mac, 111
- atcab\_verify\_validate, 112
- atcab\_version, 112
- atcab\_wakeup, 113
- atcab\_write, 113
- atcab\_write\_bytes\_zone, 113
- atcab\_write\_config\_counter, 114
- atcab\_write\_config\_zone, 114
- atcab\_write\_enc, 115
- atcab\_write\_pubkey, 115
- atcab\_write\_zone, 116
- calib\_aes\_gcm\_aad\_update, 116
- calib\_aes\_gcm\_decrypt\_finish, 117
- calib\_aes\_gcm\_decrypt\_update, 117
- calib\_aes\_gcm\_encrypt\_finish, 118
- calib\_aes\_gcm\_encrypt\_update, 118
- calib\_aes\_gcm\_init, 119
- calib\_aes\_gcm\_init\_rand, 119
- SHA\_CONTEXT\_MAX\_SIZE, 59
- Basic Crypto API methods for CryptoAuth Devices
  - (calib\_), 201
  - \_calib\_exit, 206
  - atca\_basic\_aes\_gcm\_version, 250
  - atca\_hmac\_sha256\_ctx\_t, 206
  - atca\_sha256\_ctx\_t, 206
  - calib\_aes, 207
  - calib\_aes\_decrypt, 207
  - calib\_aes\_encrypt, 208
  - calib\_aes\_gfm, 208
  - calib\_cfg\_discover, 209
  - calib\_challenge, 209
  - calib\_challenge\_seed\_update, 209
  - calib\_checkmac, 210
  - calib\_cmp\_config\_zone, 210
  - calib\_counter, 211
  - calib\_counter\_increment, 211
  - calib\_counter\_read, 212
  - calib\_derivekey, 212
  - calib\_ecdh, 213
  - calib\_ecdh\_base, 213
  - calib\_ecdh\_enc, 214
  - calib\_ecdh\_ioenc, 214
  - calib\_ecdh\_tempkey, 215
  - calib\_ecdh\_tempkey\_ioenc, 215
  - calib\_gendig, 216
  - calib\_genkey, 216
  - calib\_genkey\_base, 216
  - calib\_get\_addr, 217
  - calib\_get\_pubkey, 217
  - calib\_get\_zone\_size, 218
  - calib\_hmac, 218
  - calib\_hw\_sha2\_256, 219
  - calib\_hw\_sha2\_256\_finish, 219
  - calib\_hw\_sha2\_256\_init, 220
  - calib\_hw\_sha2\_256\_update, 220
  - calib\_idle, 221
  - calib\_info, 221
  - calib\_info\_base, 221
  - calib\_info\_get\_latch, 222
  - calib\_info\_set\_latch, 222
  - calib\_is\_locked, 223
  - calib\_is\_slot\_locked, 223
  - calib\_kdf, 223
  - calib\_lock, 224
  - calib\_lock\_config\_zone, 225
  - calib\_lock\_config\_zone\_crc, 225
  - calib\_lock\_data\_slot, 225
  - calib\_lock\_data\_zone, 226
  - calib\_lock\_data\_zone\_crc, 226
  - calib\_mac, 227
  - calib\_nonce, 227
  - calib\_nonce\_base, 227
  - calib\_nonce\_load, 228
  - calib\_nonce\_rand, 229
  - calib\_priv\_write, 229
  - calib\_random, 229
  - calib\_read\_bytes\_zone, 230
  - calib\_read\_config\_zone, 230
  - calib\_read\_enc, 231
  - calib\_read\_pubkey, 231
  - calib\_read\_serial\_number, 231
  - calib\_read\_sig, 232
  - calib\_read\_zone, 232

- calib\_secureboot, [233](#)
- calib\_secureboot\_mac, [233](#)
- calib\_selftest, [234](#)
- calib\_sha, [235](#)
- calib\_sha\_base, [235](#)
- calib\_sha\_end, [236](#)
- calib\_sha\_hmac, [236](#)
- calib\_sha\_hmac\_finish, [237](#)
- calib\_sha\_hmac\_init, [237](#)
- calib\_sha\_hmac\_update, [238](#)
- calib\_sha\_read\_context, [238](#)
- calib\_sha\_start, [239](#)
- calib\_sha\_update, [239](#)
- calib\_sha\_write\_context, [239](#)
- calib\_sign, [240](#)
- calib\_sign\_base, [240](#)
- calib\_sign\_internal, [241](#)
- calib\_sleep, [241](#)
- calib\_updateextra, [242](#)
- calib\_verify, [242](#)
- calib\_verify\_extern, [243](#)
- calib\_verify\_extern\_mac, [243](#)
- calib\_verify\_invalidate, [244](#)
- calib\_verify\_stored, [245](#)
- calib\_verify\_stored\_mac, [245](#)
- calib\_verify\_validate, [246](#)
- calib\_wakeup, [246](#)
- calib\_write, [247](#)
- calib\_write\_bytes\_zone, [247](#)
- calib\_write\_config\_counter, [248](#)
- calib\_write\_config\_zone, [248](#)
- calib\_write\_enc, [249](#)
- calib\_write\_pubkey, [249](#)
- calib\_write\_zone, [249](#)
- baud
  - ATCAIfaceCfg, [461](#)
- bBC
  - CK\_KEY\_WRAP\_SET\_OAEP\_PARAMS, [491](#)
- bind\_host\_and\_secure\_element\_with\_io\_protection
  - secure\_boot.c, [1064](#)
  - secure\_boot.h, [1066](#)
- blsExport
  - CK\_SSL3\_KEY\_MAT\_PARAMS, [513](#)
  - CK\_TLS12\_KEY\_MAT\_PARAMS, [516](#)
  - CK\_WTLS\_KEY\_MAT\_PARAMS, [525](#)
- block
  - atca\_aes\_cmac\_ctx, [408](#)
  - atca\_sha256\_ctx, [435](#)
  - hw\_sha256\_ctx, [535](#)
  - sw\_sha256\_ctx, [541](#)
- block\_size
  - atca\_aes\_cmac\_ctx, [408](#)
  - atca\_sha256\_ctx, [435](#)
  - hw\_sha256\_ctx, [535](#)
  - sw\_sha256\_ctx, [541](#)
- buf
  - atca\_jwt\_t, [430](#)
  - CL\_HashContext, [534](#)
- buflen
  - atca\_jwt\_t, [430](#)
- bus
  - ATCAIfaceCfg, [461](#)
- bus\_index
  - atcal2Cmaster, [458](#)
  - atcaSWImaster, [466](#)
- byteCount
  - CL\_HashContext, [534](#)
- byteCountHi
  - CL\_HashContext, [534](#)
- C\_CancelFunction
  - Attributes (pkcs11\_attrib\_), [343](#)
- C\_CloseAllSessions
  - Attributes (pkcs11\_attrib\_), [343](#)
- C\_CloseSession
  - Attributes (pkcs11\_attrib\_), [343](#)
- C\_CopyObject
  - Attributes (pkcs11\_attrib\_), [343](#)
- C\_CreateObject
  - Attributes (pkcs11\_attrib\_), [343](#)
- C\_Decrypt
  - Attributes (pkcs11\_attrib\_), [344](#)
- C\_DecryptDigestUpdate
  - Attributes (pkcs11\_attrib\_), [344](#)
- C\_DecryptFinal
  - Attributes (pkcs11\_attrib\_), [344](#)
- C\_DecryptInit
  - Attributes (pkcs11\_attrib\_), [344](#)
- C\_DecryptUpdate
  - Attributes (pkcs11\_attrib\_), [345](#)
- C\_DecryptVerifyUpdate
  - Attributes (pkcs11\_attrib\_), [345](#)
- C\_DeriveKey
  - Attributes (pkcs11\_attrib\_), [345](#)
- C\_DestroyObject
  - Attributes (pkcs11\_attrib\_), [345](#)
- C\_Digest
  - Attributes (pkcs11\_attrib\_), [346](#)
- C\_DigestEncryptUpdate
  - Attributes (pkcs11\_attrib\_), [346](#)
- C\_DigestFinal
  - Attributes (pkcs11\_attrib\_), [346](#)
- C\_DigestInit
  - Attributes (pkcs11\_attrib\_), [346](#)
- C\_DigestKey
  - Attributes (pkcs11\_attrib\_), [347](#)
- C\_DigestUpdate
  - Attributes (pkcs11\_attrib\_), [347](#)
- C\_Encrypt
  - Attributes (pkcs11\_attrib\_), [347](#)
- C\_EncryptFinal
  - Attributes (pkcs11\_attrib\_), [347](#)
- C\_EncryptInit
  - Attributes (pkcs11\_attrib\_), [347](#)
- C\_EncryptUpdate
  - Attributes (pkcs11\_attrib\_), [348](#)
- C\_Finalize

- Attributes (pkcs11\_attrib\_), 348
- C\_FindObjects
  - Attributes (pkcs11\_attrib\_), 348
- C\_FindObjectsFinal
  - Attributes (pkcs11\_attrib\_), 348
- C\_FindObjectsInit
  - Attributes (pkcs11\_attrib\_), 348
- C\_GenerateKey
  - Attributes (pkcs11\_attrib\_), 349
- C\_GenerateKeyPair
  - Attributes (pkcs11\_attrib\_), 349
- C\_GenerateRandom
  - Attributes (pkcs11\_attrib\_), 349
- C\_GetAttributeValue
  - Attributes (pkcs11\_attrib\_), 349
- C\_GetFunctionList
  - Attributes (pkcs11\_attrib\_), 350
- C\_GetFunctionStatus
  - Attributes (pkcs11\_attrib\_), 350
- C\_GetInfo
  - Attributes (pkcs11\_attrib\_), 350
- C\_GetMechanismInfo
  - Attributes (pkcs11\_attrib\_), 350
- C\_GetMechanismList
  - Attributes (pkcs11\_attrib\_), 350
- C\_GetObjectSize
  - Attributes (pkcs11\_attrib\_), 351
- C\_GetOperationState
  - Attributes (pkcs11\_attrib\_), 351
- C\_GetSessionInfo
  - Attributes (pkcs11\_attrib\_), 351
- C\_GetSlotInfo
  - Attributes (pkcs11\_attrib\_), 351
- C\_GetSlotList
  - Attributes (pkcs11\_attrib\_), 351
- C\_GetTokenInfo
  - Attributes (pkcs11\_attrib\_), 352
- C\_Initialize
  - Attributes (pkcs11\_attrib\_), 352
- C\_InitPIN
  - Attributes (pkcs11\_attrib\_), 352
- C\_InitToken
  - Attributes (pkcs11\_attrib\_), 352
- C\_Login
  - Attributes (pkcs11\_attrib\_), 352
- C\_Logout
  - Attributes (pkcs11\_attrib\_), 353
- C\_OpenSession
  - Attributes (pkcs11\_attrib\_), 353
- C\_SeedRandom
  - Attributes (pkcs11\_attrib\_), 353
- C\_SetAttributeValue
  - Attributes (pkcs11\_attrib\_), 353
- C\_SetOperationState
  - Attributes (pkcs11\_attrib\_), 354
- C\_SetPIN
  - Attributes (pkcs11\_attrib\_), 354
- C\_Sign
  - Attributes (pkcs11\_attrib\_), 354
- C\_SignEncryptUpdate
  - Attributes (pkcs11\_attrib\_), 354
- C\_SignFinal
  - Attributes (pkcs11\_attrib\_), 355
- C\_SignInit
  - Attributes (pkcs11\_attrib\_), 355
- C\_SignRecover
  - Attributes (pkcs11\_attrib\_), 355
- C\_SignRecoverInit
  - Attributes (pkcs11\_attrib\_), 355
- C\_SignUpdate
  - Attributes (pkcs11\_attrib\_), 356
- C\_UnwrapKey
  - Attributes (pkcs11\_attrib\_), 356
- C\_Verify
  - Attributes (pkcs11\_attrib\_), 356
- C\_VerifyFinal
  - Attributes (pkcs11\_attrib\_), 356
- C\_VerifyInit
  - Attributes (pkcs11\_attrib\_), 357
- C\_VerifyRecover
  - Attributes (pkcs11\_attrib\_), 357
- C\_VerifyRecoverInit
  - Attributes (pkcs11\_attrib\_), 357
- C\_VerifyUpdate
  - Attributes (pkcs11\_attrib\_), 357
- C\_WaitForSlotEvent
  - Attributes (pkcs11\_attrib\_), 357
- C\_WrapKey
  - Attributes (pkcs11\_attrib\_), 358
- ca\_cert\_def
  - atcacert\_def\_s, 449
- calib\_aes
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 207
- calib\_aes.c, 672
- calib\_aes\_decrypt
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 207
- calib\_aes\_encrypt
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 208
- calib\_aes\_gcm.c, 673
  - calib\_aes\_gcm\_aad\_update, 674
  - calib\_aes\_gcm\_decrypt\_finish, 674
  - calib\_aes\_gcm\_decrypt\_update, 675
  - calib\_aes\_gcm\_encrypt\_finish, 675
  - calib\_aes\_gcm\_encrypt\_update, 676
  - calib\_aes\_gcm\_init, 676
  - calib\_aes\_gcm\_init\_rand, 677
  - RETURN, 674
- calib\_aes\_gcm.h, 677
- calib\_aes\_gcm\_aad\_update
  - Basic Crypto API methods (atcab\_), 116
- calib\_aes\_gcm.c, 674
- calib\_aes\_gcm\_decrypt\_finish
  - Basic Crypto API methods (atcab\_), 117

- calib\_aes\_gcm.c, 674
- calib\_aes\_gcm\_decrypt\_update
  - Basic Crypto API methods (atcab\_), 117
  - calib\_aes\_gcm.c, 675
- calib\_aes\_gcm\_encrypt\_finish
  - Basic Crypto API methods (atcab\_), 118
  - calib\_aes\_gcm.c, 675
- calib\_aes\_gcm\_encrypt\_update
  - Basic Crypto API methods (atcab\_), 118
  - calib\_aes\_gcm.c, 676
- calib\_aes\_gcm\_init
  - Basic Crypto API methods (atcab\_), 119
  - calib\_aes\_gcm.c, 676
- calib\_aes\_gcm\_init\_rand
  - Basic Crypto API methods (atcab\_), 119
  - calib\_aes\_gcm.c, 677
- calib\_aes\_gfm
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 208
- calib\_basic.c, 678
  - MAX\_BUSES, 679
- calib\_basic.h, 679
- calib\_cfg\_discover
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 209
- calib\_challenge
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 209
- calib\_challenge\_seed\_update
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 209
- calib\_checkmac
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 210
- calib\_checkmac.c, 685
- calib\_cmp\_config\_zone
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 210
- calib\_command.c, 685
  - atAES, 687
  - atCalcCrc, 687
  - atCheckCrc, 688
  - atCheckMAC, 688
  - atCounter, 688
  - atCRC, 689
  - atDeriveKey, 689
  - atECDH, 690
  - atGenDig, 690
  - atGenKey, 690
  - atHMAC, 691
  - atInfo, 691
  - atIsECCFamily, 691
  - atIsSHAFamily, 692
  - atKDF, 692
  - atLock, 693
  - atMAC, 693
  - atNonce, 693
  - atPause, 694
  - atPrivWrite, 694
  - atRandom, 694
  - atRead, 695
  - atSecureBoot, 695
  - atSelfTest, 696
  - atSHA, 696
  - atSign, 696
  - atUpdateExtra, 697
  - atVerify, 697
  - atWrite, 698
  - isATCAError, 698
- calib\_command.h, 698
  - AES\_COUNT, 717
  - AES\_DATA\_SIZE, 717
  - AES\_INPUT\_IDX, 717
  - AES\_KEYID\_IDX, 717
  - AES\_MODE\_DECRYPT, 717
  - AES\_MODE\_ENCRYPT, 717
  - AES\_MODE\_GFM, 718
  - AES\_MODE\_IDX, 718
  - AES\_MODE\_KEY\_BLOCK\_MASK, 718
  - AES\_MODE\_KEY\_BLOCK\_POS, 718
  - AES\_MODE\_MASK, 718
  - AES\_MODE\_OP\_MASK, 718
  - AES\_RSP\_SIZE, 719
  - atAES, 788
  - ATCA\_ADDRESS\_MASK, 719
  - ATCA\_ADDRESS\_MASK\_CONFIG, 719
  - ATCA\_ADDRESS\_MASK\_OTP, 719
  - ATCA\_AES, 719
  - ATCA\_AES\_GFM\_SIZE, 719
  - ATCA\_AES\_KEY\_TYPE, 720
  - ATCA\_B283\_KEY\_TYPE, 720
  - ATCA\_BLOCK\_SIZE, 720
  - ATCA\_CHECKMAC, 720
  - ATCA\_CHIPMODE\_CLOCK\_DIV\_M0, 720
  - ATCA\_CHIPMODE\_CLOCK\_DIV\_M1, 720
  - ATCA\_CHIPMODE\_CLOCK\_DIV\_M2, 721
  - ATCA\_CHIPMODE\_CLOCK\_DIV\_MASK, 721
  - ATCA\_CHIPMODE\_I2C\_ADDRESS\_FLAG, 721
  - ATCA\_CHIPMODE\_OFFSET, 721
  - ATCA\_CHIPMODE\_TTL\_ENABLE\_FLAG, 721
  - ATCA\_CHIPMODE\_WATCHDOG\_LONG, 721
  - ATCA\_CHIPMODE\_WATCHDOG\_MASK, 722
  - ATCA\_CHIPMODE\_WATCHDOG\_SHORT, 722
  - ATCA\_CMD\_SIZE\_MAX, 722
  - ATCA\_CMD\_SIZE\_MIN, 722
  - ATCA\_COUNT\_IDX, 722
  - ATCA\_COUNT\_SIZE, 722
  - ATCA\_COUNTER, 723
  - ATCA\_CRC\_SIZE, 723
  - ATCA\_DATA\_IDX, 723
  - ATCA\_DATA\_SIZE, 723
  - ATCA\_DERIVE\_KEY, 723
  - ATCA\_ECC\_CONFIG\_SIZE, 723
  - ATCA\_ECDH, 724
  - ATCA\_GENDIG, 724
  - ATCA\_GENKEY, 724

ATCA\_HMAC, 724  
 ATCA\_INFO, 724  
 ATCA\_K283\_KEY\_TYPE, 724  
 ATCA\_KDF, 725  
 ATCA\_KEY\_COUNT, 725  
 ATCA\_KEY\_ID\_MAX, 725  
 ATCA\_KEY\_SIZE, 725  
 ATCA\_LOCK, 725  
 ATCA\_LOCKED, 725  
 ATCA\_MAC, 726  
 ATCA\_NONCE, 726  
 ATCA\_OPCODE\_IDX, 726  
 ATCA\_OTP\_BLOCK\_MAX, 726  
 ATCA\_OTP\_SIZE, 726  
 ATCA\_P256\_KEY\_TYPE, 726  
 ATCA\_PACKET\_OVERHEAD, 727  
 ATCA\_PARAM1\_IDX, 727  
 ATCA\_PARAM2\_IDX, 727  
 ATCA\_PAUSE, 727  
 ATCA\_PRIV\_KEY\_SIZE, 727  
 ATCA\_PRIVWRITE, 727  
 ATCA\_PUB\_KEY\_PAD, 728  
 ATCA\_PUB\_KEY\_SIZE, 728  
 ATCA\_RANDOM, 728  
 ATCA\_READ, 728  
 ATCA\_RSP\_DATA\_IDX, 728  
 ATCA\_RSP\_SIZE\_16, 728  
 ATCA\_RSP\_SIZE\_32, 729  
 ATCA\_RSP\_SIZE\_4, 729  
 ATCA\_RSP\_SIZE\_64, 729  
 ATCA\_RSP\_SIZE\_72, 729  
 ATCA\_RSP\_SIZE\_MAX, 729  
 ATCA\_RSP\_SIZE\_MIN, 729  
 ATCA\_RSP\_SIZE\_VAL, 730  
 ATCA\_SECUREBOOT, 730  
 ATCA\_SELFTEST, 730  
 ATCA\_SERIAL\_NUM\_SIZE, 730  
 ATCA\_SHA, 730  
 ATCA\_SHA\_CONFIG\_SIZE, 730  
 ATCA\_SHA\_DIGEST\_SIZE, 731  
 ATCA\_SHA\_KEY\_TYPE, 731  
 ATCA\_SIG\_SIZE, 731  
 ATCA\_SIGN, 731  
 ATCA\_TEMPKEY\_KEYID, 731  
 ATCA\_UNLOCKED, 731  
 ATCA\_UPDATE\_EXTRA, 732  
 ATCA\_VERIFY, 732  
 ATCA\_WORD\_SIZE, 732  
 ATCA\_WRITE, 732  
 ATCA\_ZONE\_ENCRYPTED, 732  
 ATCA\_ZONE\_MASK, 732  
 ATCA\_ZONE\_READWRITE\_32, 733  
 atCalcCrc, 789  
 atCheckCrc, 789  
 atCheckMAC, 789  
 atCounter, 790  
 atCRC, 790  
 atDeriveKey, 790  
 atECDH, 791  
 atGenDig, 791  
 atGenKey, 792  
 atHMAC, 792  
 atInfo, 792  
 atIsECCFamily, 793  
 atIsSHAFamily, 793  
 atKDF, 794  
 atLock, 794  
 atMAC, 794  
 atNonce, 795  
 atPause, 795  
 atPrivWrite, 795  
 atRandom, 796  
 atRead, 796  
 atSecureBoot, 797  
 atSelfTest, 797  
 atSHA, 797  
 atSign, 799  
 atUpdateExtra, 799  
 atVerify, 799  
 atWrite, 800  
 CHECKMAC\_CLIENT\_CHALLENGE\_IDX, 733  
 CHECKMAC\_CLIENT\_CHALLENGE\_SIZE, 733  
 CHECKMAC\_CLIENT\_COMMAND\_SIZE, 733  
 CHECKMAC\_CLIENT\_RESPONSE\_IDX, 733  
 CHECKMAC\_CLIENT\_RESPONSE\_SIZE, 733  
 CHECKMAC\_CMD\_MATCH, 734  
 CHECKMAC\_CMD\_MISMATCH, 734  
 CHECKMAC\_COUNT, 734  
 CHECKMAC\_DATA\_IDX, 734  
 CHECKMAC\_KEYID\_IDX, 734  
 CHECKMAC\_MODE\_BLOCK1\_TEMPKEY, 734  
 CHECKMAC\_MODE\_BLOCK2\_TEMPKEY, 735  
 CHECKMAC\_MODE\_CHALLENGE, 735  
 CHECKMAC\_MODE\_IDX, 735  
 CHECKMAC\_MODE\_INCLUDE\_OTP\_64, 735  
 CHECKMAC\_MODE\_MASK, 735  
 CHECKMAC\_MODE\_SOURCE\_FLAG\_MATCH, 735  
 CHECKMAC\_OTHER\_DATA\_SIZE, 736  
 CHECKMAC\_RSP\_SIZE, 736  
 CMD\_STATUS\_BYTE\_COMM, 736  
 CMD\_STATUS\_BYTE\_ECC, 736  
 CMD\_STATUS\_BYTE\_EXEC, 736  
 CMD\_STATUS\_BYTE\_PARSE, 736  
 CMD\_STATUS\_SUCCESS, 737  
 CMD\_STATUS\_WAKEUP, 737  
 COUNTER\_COUNT, 737  
 COUNTER\_KEYID\_IDX, 737  
 COUNTER\_MAX\_VALUE, 737  
 COUNTER\_MODE\_IDX, 737  
 COUNTER\_MODE\_INCREMENT, 738  
 COUNTER\_MODE\_MASK, 738  
 COUNTER\_MODE\_READ, 738  
 COUNTER\_RSP\_SIZE, 738  
 COUNTER\_SIZE, 738  
 DERIVE\_KEY\_COUNT\_LARGE, 738



DERIVE\_KEY\_COUNT\_SMALL, 739  
DERIVE\_KEY\_MAC\_IDX, 739  
DERIVE\_KEY\_MAC\_SIZE, 739  
DERIVE\_KEY\_MODE, 739  
DERIVE\_KEY\_RANDOM\_FLAG, 739  
DERIVE\_KEY\_RANDOM\_IDX, 739  
DERIVE\_KEY\_RSP\_SIZE, 740  
DERIVE\_KEY\_TARGETKEY\_IDX, 740  
ECDH\_COUNT, 740  
ECDH\_KEY\_SIZE, 740  
ECDH\_MODE\_COPY\_COMPATIBLE, 740  
ECDH\_MODE\_COPY\_EEPROM\_SLOT, 740  
ECDH\_MODE\_COPY\_MASK, 740  
ECDH\_MODE\_COPY\_OUTPUT\_BUFFER, 741  
ECDH\_MODE\_COPY\_TEMP\_KEY, 741  
ECDH\_MODE\_OUTPUT\_CLEAR, 741  
ECDH\_MODE\_OUTPUT\_ENC, 741  
ECDH\_MODE\_OUTPUT\_MASK, 741  
ECDH\_MODE\_SOURCE\_EEPROM\_SLOT, 741  
ECDH\_MODE\_SOURCE\_MASK, 741  
ECDH\_MODE\_SOURCE\_TEMPKEY, 741  
ECDH\_PREFIX\_MODE, 742  
ECDH\_RSP\_SIZE, 742  
GENDIG\_COUNT, 742  
GENDIG\_DATA\_IDX, 742  
GENDIG\_KEYID\_IDX, 742  
GENDIG\_RSP\_SIZE, 742  
GENDIG\_ZONE\_CONFIG, 743  
GENDIG\_ZONE\_COUNTER, 743  
GENDIG\_ZONE\_DATA, 743  
GENDIG\_ZONE\_IDX, 743  
GENDIG\_ZONE\_KEY\_CONFIG, 743  
GENDIG\_ZONE\_OTP, 743  
GENDIG\_ZONE\_SHARED\_NONCE, 744  
GENKEY\_COUNT, 744  
GENKEY\_COUNT\_DATA, 744  
GENKEY\_DATA\_IDX, 744  
GENKEY\_KEYID\_IDX, 744  
GENKEY\_MODE\_DIGEST, 744  
GENKEY\_MODE\_IDX, 745  
GENKEY\_MODE\_MASK, 745  
GENKEY\_MODE\_PRIVATE, 745  
GENKEY\_MODE\_PUBKEY\_DIGEST, 745  
GENKEY\_MODE\_PUBLIC, 745  
GENKEY\_OTHER\_DATA\_SIZE, 745  
GENKEY\_PRIVATE\_TO\_TEMPKEY, 746  
GENKEY\_RSP\_SIZE\_LONG, 746  
GENKEY\_RSP\_SIZE\_SHORT, 746  
HMAC\_COUNT, 746  
HMAC\_DIGEST\_SIZE, 746  
HMAC\_KEYID\_IDX, 746  
HMAC\_MODE\_FLAG\_FULLSN, 747  
HMAC\_MODE\_FLAG\_OTP64, 747  
HMAC\_MODE\_FLAG\_OTP88, 747  
HMAC\_MODE\_FLAG\_TK\_NORAND, 747  
HMAC\_MODE\_FLAG\_TK\_RAND, 747  
HMAC\_MODE\_IDX, 747  
HMAC\_MODE\_MASK, 748  
HMAC\_RSP\_SIZE, 748  
INFO\_COUNT, 748  
INFO\_DRIVER\_STATE\_MASK, 748  
INFO\_MODE\_GPIO, 748  
INFO\_MODE\_KEY\_VALID, 748  
INFO\_MODE\_MAX, 749  
INFO\_MODE\_REVISION, 749  
INFO\_MODE\_STATE, 749  
INFO\_MODE\_VOL\_KEY\_PERMIT, 749  
INFO\_NO\_STATE, 749  
INFO\_OUTPUT\_STATE\_MASK, 749  
INFO\_PARAM1\_IDX, 750  
INFO\_PARAM2\_IDX, 750  
INFO\_PARAM2\_LATCH\_CLEAR, 750  
INFO\_PARAM2\_LATCH\_SET, 750  
INFO\_PARAM2\_SET\_LATCH\_STATE, 750  
INFO\_RSP\_SIZE, 750  
INFO\_SIZE, 751  
isATCAError, 800  
KDF\_DETAILS\_AES\_KEY\_LOC\_MASK, 751  
KDF\_DETAILS\_HKDF\_MSG\_LOC\_INPUT, 751  
KDF\_DETAILS\_HKDF\_MSG\_LOC\_IV, 751  
KDF\_DETAILS\_HKDF\_MSG\_LOC\_MASK, 751  
KDF\_DETAILS\_HKDF\_MSG\_LOC\_SLOT, 751  
KDF\_DETAILS\_HKDF\_MSG\_LOC\_TEMPKEY, 752  
KDF\_DETAILS\_HKDF\_ZERO\_KEY, 752  
KDF\_DETAILS\_IDX, 752  
KDF\_DETAILS\_PRF\_AEAD\_MASK, 752  
KDF\_DETAILS\_PRF\_AEAD\_MODE0, 752  
KDF\_DETAILS\_PRF\_AEAD\_MODE1, 752  
KDF\_DETAILS\_PRF\_KEY\_LEN\_16, 753  
KDF\_DETAILS\_PRF\_KEY\_LEN\_32, 753  
KDF\_DETAILS\_PRF\_KEY\_LEN\_48, 753  
KDF\_DETAILS\_PRF\_KEY\_LEN\_64, 753  
KDF\_DETAILS\_PRF\_KEY\_LEN\_MASK, 753  
KDF\_DETAILS\_PRF\_TARGET\_LEN\_32, 753  
KDF\_DETAILS\_PRF\_TARGET\_LEN\_64, 754  
KDF\_DETAILS\_PRF\_TARGET\_LEN\_MASK, 754  
KDF\_DETAILS\_SIZE, 754  
KDF\_KEYID\_IDX, 754  
KDF\_MESSAGE\_IDX, 754  
KDF\_MODE\_ALG\_AES, 754  
KDF\_MODE\_ALG\_HKDF, 755  
KDF\_MODE\_ALG\_MASK, 755  
KDF\_MODE\_ALG\_PRF, 755  
KDF\_MODE\_IDX, 755  
KDF\_MODE\_SOURCE\_ALTKEYBUF, 755  
KDF\_MODE\_SOURCE\_MASK, 755  
KDF\_MODE\_SOURCE\_SLOT, 756  
KDF\_MODE\_SOURCE\_TEMPKEY, 756  
KDF\_MODE\_SOURCE\_TEMPKEY\_UP, 756  
KDF\_MODE\_TARGET\_ALTKEYBUF, 756  
KDF\_MODE\_TARGET\_MASK, 756  
KDF\_MODE\_TARGET\_OUTPUT, 756  
KDF\_MODE\_TARGET\_OUTPUT\_ENC, 757  
KDF\_MODE\_TARGET\_SLOT, 757  
KDF\_MODE\_TARGET\_TEMPKEY, 757

KDF\_MODE\_TARGET\_TEMPKEY\_UP, [757](#)  
LOCK\_COUNT, [757](#)  
LOCK\_RSP\_SIZE, [757](#)  
LOCK\_SUMMARY\_IDX, [758](#)  
LOCK\_ZONE\_CONFIG, [758](#)  
LOCK\_ZONE\_DATA, [758](#)  
LOCK\_ZONE\_DATA\_SLOT, [758](#)  
LOCK\_ZONE\_IDX, [758](#)  
LOCK\_ZONE\_MASK, [758](#)  
LOCK\_ZONE\_NO\_CRC, [759](#)  
MAC\_CHALLENGE\_IDX, [759](#)  
MAC\_CHALLENGE\_SIZE, [759](#)  
MAC\_COUNT\_LONG, [759](#)  
MAC\_COUNT\_SHORT, [759](#)  
MAC\_KEYID\_IDX, [759](#)  
MAC\_MODE\_BLOCK1\_TEMPKEY, [760](#)  
MAC\_MODE\_BLOCK2\_TEMPKEY, [760](#)  
MAC\_MODE\_CHALLENGE, [760](#)  
MAC\_MODE\_IDX, [760](#)  
MAC\_MODE\_INCLUDE\_OTP\_64, [760](#)  
MAC\_MODE\_INCLUDE\_OTP\_88, [760](#)  
MAC\_MODE\_INCLUDE\_SN, [761](#)  
MAC\_MODE\_MASK, [761](#)  
MAC\_MODE\_PASSTHROUGH, [761](#)  
MAC\_MODE\_PTNONCE\_TEMPKEY, [761](#)  
MAC\_MODE\_SOURCE\_FLAG\_MATCH, [761](#)  
MAC\_RSP\_SIZE, [761](#)  
MAC\_SIZE, [762](#)  
NONCE\_COUNT\_LONG, [762](#)  
NONCE\_COUNT\_LONG\_64, [762](#)  
NONCE\_COUNT\_SHORT, [762](#)  
NONCE\_INPUT\_IDX, [762](#)  
NONCE\_MODE\_IDX, [762](#)  
NONCE\_MODE\_INPUT\_LEN\_32, [763](#)  
NONCE\_MODE\_INPUT\_LEN\_64, [763](#)  
NONCE\_MODE\_INPUT\_LEN\_MASK, [763](#)  
NONCE\_MODE\_INVALID, [763](#)  
NONCE\_MODE\_MASK, [763](#)  
NONCE\_MODE\_NO\_SEED\_UPDATE, [763](#)  
NONCE\_MODE\_PASSTHROUGH, [764](#)  
NONCE\_MODE\_SEED\_UPDATE, [764](#)  
NONCE\_MODE\_TARGET\_ALTKEYBUF, [764](#)  
NONCE\_MODE\_TARGET\_MASK, [764](#)  
NONCE\_MODE\_TARGET\_MSGDIGBUF, [764](#)  
NONCE\_MODE\_TARGET\_TEMPKEY, [764](#)  
NONCE\_NUMIN\_SIZE, [765](#)  
NONCE\_NUMIN\_SIZE\_PASSTHROUGH, [765](#)  
NONCE\_PARAM2\_IDX, [765](#)  
NONCE\_RSP\_SIZE\_LONG, [765](#)  
NONCE\_RSP\_SIZE\_SHORT, [765](#)  
NONCE\_ZERO\_CALC\_MASK, [765](#)  
NONCE\_ZERO\_CALC\_RANDOM, [766](#)  
NONCE\_ZERO\_CALC\_TEMPKEY, [766](#)  
OUTNONCE\_SIZE, [766](#)  
PAUSE\_COUNT, [766](#)  
PAUSE\_PARAM2\_IDX, [766](#)  
PAUSE\_RSP\_SIZE, [766](#)  
PAUSE\_SELECT\_IDX, [767](#)  
PRIVWRITE\_COUNT, [767](#)  
PRIVWRITE\_KEYID\_IDX, [767](#)  
PRIVWRITE\_MAC\_IDX, [767](#)  
PRIVWRITE\_MODE\_ENCRYPT, [767](#)  
PRIVWRITE\_RSP\_SIZE, [767](#)  
PRIVWRITE\_VALUE\_IDX, [768](#)  
PRIVWRITE\_ZONE\_IDX, [768](#)  
PRIVWRITE\_ZONE\_MASK, [768](#)  
RANDOM\_COUNT, [768](#)  
RANDOM\_MODE\_IDX, [768](#)  
RANDOM\_NO\_SEED\_UPDATE, [768](#)  
RANDOM\_NUM\_SIZE, [769](#)  
RANDOM\_PARAM2\_IDX, [769](#)  
RANDOM\_RSP\_SIZE, [769](#)  
RANDOM\_SEED\_UPDATE, [769](#)  
READ\_32\_RSP\_SIZE, [769](#)  
READ\_4\_RSP\_SIZE, [769](#)  
READ\_ADDR\_IDX, [770](#)  
READ\_COUNT, [770](#)  
READ\_ZONE\_IDX, [770](#)  
READ\_ZONE\_MASK, [770](#)  
RSA2048\_KEY\_SIZE, [770](#)  
SECUREBOOT\_COUNT\_DIG, [770](#)  
SECUREBOOT\_COUNT\_DIG\_SIG, [771](#)  
SECUREBOOT\_DIGEST\_SIZE, [771](#)  
SECUREBOOT\_MAC\_SIZE, [771](#)  
SECUREBOOT\_MODE\_ENC\_MAC\_FLAG, [771](#)  
SECUREBOOT\_MODE\_FULL, [771](#)  
SECUREBOOT\_MODE\_FULL\_COPY, [771](#)  
SECUREBOOT\_MODE\_FULL\_STORE, [772](#)  
SECUREBOOT\_MODE\_IDX, [772](#)  
SECUREBOOT\_MODE\_MASK, [772](#)  
SECUREBOOT\_MODE\_PROHIBIT\_FLAG, [772](#)  
SECUREBOOT\_RSP\_SIZE\_MAC, [772](#)  
SECUREBOOT\_RSP\_SIZE\_NO\_MAC, [772](#)  
SECUREBOOT\_SIGNATURE\_SIZE, [773](#)  
SECUREBOOTCONFIG\_MODE\_DISABLED, [773](#)  
SECUREBOOTCONFIG\_MODE\_FULL\_BOTH, [773](#)  
SECUREBOOTCONFIG\_MODE\_FULL\_DIG, [773](#)  
SECUREBOOTCONFIG\_MODE\_FULL\_SIG, [773](#)  
SECUREBOOTCONFIG\_MODE\_MASK, [773](#)  
SECUREBOOTCONFIG\_OFFSET, [774](#)  
SELFTEST\_COUNT, [774](#)  
SELFTEST\_MODE\_AES, [774](#)  
SELFTEST\_MODE\_ALL, [774](#)  
SELFTEST\_MODE\_ECDH, [774](#)  
SELFTEST\_MODE\_ECDSA\_SIGN\_VERIFY, [774](#)  
SELFTEST\_MODE\_IDX, [775](#)  
SELFTEST\_MODE\_RNG, [775](#)  
SELFTEST\_MODE\_SHA, [775](#)  
SELFTEST\_RSP\_SIZE, [775](#)  
SHA\_COUNT\_LONG, [775](#)  
SHA\_COUNT\_SHORT, [775](#)  
SHA\_DATA\_MAX, [776](#)  
SHA\_MODE\_608\_HMAC\_END, [776](#)  
SHA\_MODE\_HMAC\_END, [776](#)  
SHA\_MODE\_HMAC\_START, [776](#)



- SHA\_MODE\_HMAC\_UPDATE, 776
- SHA\_MODE\_MASK, 776
- SHA\_MODE\_READ\_CONTEXT, 777
- SHA\_MODE\_SHA256\_END, 777
- SHA\_MODE\_SHA256\_PUBLIC, 777
- SHA\_MODE\_SHA256\_START, 777
- SHA\_MODE\_SHA256\_UPDATE, 777
- SHA\_MODE\_TARGET\_MASK, 777
- SHA\_MODE\_WRITE\_CONTEXT, 778
- SHA\_RSP\_SIZE, 778
- SHA\_RSP\_SIZE\_LONG, 778
- SHA\_RSP\_SIZE\_SHORT, 778
- SIGN\_COUNT, 778
- SIGN\_KEYID\_IDX, 778
- SIGN\_MODE\_EXTERNAL, 779
- SIGN\_MODE\_IDX, 779
- SIGN\_MODE\_INCLUDE\_SN, 779
- SIGN\_MODE\_INTERNAL, 779
- SIGN\_MODE\_INVALIDATE, 779
- SIGN\_MODE\_MASK, 779
- SIGN\_MODE\_SOURCE\_MASK, 780
- SIGN\_MODE\_SOURCE\_MSGDIGBUF, 780
- SIGN\_MODE\_SOURCE\_TEMPKEY, 780
- SIGN\_RSP\_SIZE, 780
- UPDATE\_COUNT, 780
- UPDATE\_MODE\_DEC\_COUNTER, 780
- UPDATE\_MODE\_IDX, 781
- UPDATE\_MODE\_SELECTOR, 781
- UPDATE\_MODE\_USER\_EXTRA, 781
- UPDATE\_MODE\_USER\_EXTRA\_ADD, 781
- UPDATE\_RSP\_SIZE, 781
- UPDATE\_VALUE\_IDX, 781
- VERIFY\_256\_EXTERNAL\_COUNT, 782
- VERIFY\_256\_KEY\_SIZE, 782
- VERIFY\_256\_SIGNATURE\_SIZE, 782
- VERIFY\_256\_STORED\_COUNT, 782
- VERIFY\_256\_VALIDATE\_COUNT, 782
- VERIFY\_283\_EXTERNAL\_COUNT, 782
- VERIFY\_283\_KEY\_SIZE, 783
- VERIFY\_283\_SIGNATURE\_SIZE, 783
- VERIFY\_283\_STORED\_COUNT, 783
- VERIFY\_283\_VALIDATE\_COUNT, 783
- VERIFY\_DATA\_IDX, 783
- VERIFY\_KEY\_B283, 783
- VERIFY\_KEY\_K283, 784
- VERIFY\_KEY\_P256, 784
- VERIFY\_KEYID\_IDX, 784
- VERIFY\_MODE\_EXTERNAL, 784
- VERIFY\_MODE\_IDX, 784
- VERIFY\_MODE\_INVALIDATE, 784
- VERIFY\_MODE\_MAC\_FLAG, 785
- VERIFY\_MODE\_MASK, 785
- VERIFY\_MODE\_SOURCE\_MASK, 785
- VERIFY\_MODE\_SOURCE\_MSGDIGBUF, 785
- VERIFY\_MODE\_SOURCE\_TEMPKEY, 785
- VERIFY\_MODE\_STORED, 785
- VERIFY\_MODE\_VALIDATE, 786
- VERIFY\_MODE\_VALIDATE\_EXTERNAL, 786
- VERIFY\_OTHER\_DATA\_SIZE, 786
- VERIFY\_RSP\_SIZE, 786
- VERIFY\_RSP\_SIZE\_MAC, 786
- WRITE\_ADDR\_IDX, 786
- WRITE\_MAC\_SIZE, 787
- WRITE\_MAC\_VL\_IDX, 787
- WRITE\_MAC\_VS\_IDX, 787
- WRITE\_RSP\_SIZE, 787
- WRITE\_VALUE\_IDX, 787
- WRITE\_ZONE\_DATA, 787
- WRITE\_ZONE\_IDX, 788
- WRITE\_ZONE\_MASK, 788
- WRITE\_ZONE\_OTP, 788
- WRITE\_ZONE\_WITH\_MAC, 788
- calib\_counter
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 211
- calib\_counter.c, 801
- calib\_counter\_increment
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 211
- calib\_counter\_read
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 212
- calib\_derivekey
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 212
- calib\_derivekey.c, 801
- calib\_ecdh
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 213
- calib\_ecdh.c, 802
  - calib\_ecdh\_enc, 803
- calib\_ecdh\_base
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 213
- calib\_ecdh\_enc
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 214
  - calib\_ecdh.c, 803
- calib\_ecdh\_ioenc
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 214
- calib\_ecdh\_tempkey
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 215
- calib\_ecdh\_tempkey\_ioenc
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 215
- calib\_execute\_command
  - calib\_execution.c, 804
  - calib\_execution.h, 806
- calib\_execution.c, 804
  - calib\_execute\_command, 804
- calib\_execution.h, 805
  - ATCA\_UNSUPPORTED\_CMD, 805
  - calib\_execute\_command, 806
- calib\_gendig

- Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 216
- calib\_gendig.c, 806
- calib\_genkey
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 216
- calib\_genkey.c, 807
- calib\_genkey\_base
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 216
- calib\_get\_addr
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 217
- calib\_get\_pubkey
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 217
- calib\_get\_zone\_size
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 218
- calib\_hmac
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 218
- calib\_hmac.c, 807
- calib\_hw\_sha2\_256
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 219
- calib\_hw\_sha2\_256\_finish
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 219
- calib\_hw\_sha2\_256\_init
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 220
- calib\_hw\_sha2\_256\_update
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 220
- calib\_idle
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 221
- calib\_info
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 221
- calib\_info.c, 808
- calib\_info\_base
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 221
- calib\_info\_get\_latch
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 222
- calib\_info\_set\_latch
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 222
- calib\_is\_locked
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 223
- calib\_is\_slot\_locked
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 223
- calib\_kdf
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 223
- calib\_kdf.c, 809
- calib\_lock
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 224
- calib\_lock.c, 809
- calib\_lock\_config\_zone
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 225
- calib\_lock\_config\_zone\_crc
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 225
- calib\_lock\_data\_slot
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 225
- calib\_lock\_data\_zone
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 226
- calib\_lock\_data\_zone\_crc
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 226
- calib\_mac
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 227
- calib\_mac.c, 810
- calib\_nonce
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 227
- calib\_nonce.c, 811
- calib\_nonce\_base
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 227
- calib\_nonce\_load
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 228
- calib\_nonce\_rand
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 229
- calib\_priv\_write
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 229
- calib\_privwrite.c, 812
- calib\_privwrite.c, 811
- calib\_priv\_write, 812
- calib\_random
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 229
- calib\_random.c, 813
- calib\_read.c, 813
- calib\_read\_enc, 814
- calib\_read\_bytes\_zone
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 230
- calib\_read\_config\_zone
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 230
- calib\_read\_enc

- Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 231
- calib\_read.c, 814
- calib\_read\_pubkey
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 231
- calib\_read\_serial\_number
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 231
- calib\_read\_sig
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 232
- calib\_read\_zone
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 232
- calib\_secureboot
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 233
- calib\_secureboot.c, 815
- calib\_secureboot\_mac
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 233
- calib\_selftest
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 234
- calib\_selftest.c, 816
- calib\_sha
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 235
- calib\_sha.c, 816
- calib\_sha\_base
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 235
- calib\_sha\_end
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 236
- calib\_sha\_hmac
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 236
- calib\_sha\_hmac\_finish
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 237
- calib\_sha\_hmac\_init
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 237
- calib\_sha\_hmac\_update
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 238
- calib\_sha\_read\_context
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 238
- calib\_sha\_start
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 239
- calib\_sha\_update
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 239
- calib\_sha\_write\_context
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 239
- calib\_sign
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 240
- calib\_sign.c, 818
- calib\_sign\_base
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 240
- calib\_sign\_internal
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 241
- calib\_sleep
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 241
- calib\_updateextra
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 242
- calib\_updateextra.c, 819
- calib\_verify
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 242
- calib\_verify.c, 819
- calib\_verify\_extern
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 243
- calib\_verify\_extern\_mac
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 243
- calib\_verify\_invalidate
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 244
- calib\_verify\_stored
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 245
- calib\_verify\_stored\_mac
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 245
- calib\_verify\_validate
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 246
- calib\_wakeup
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 246
- calib\_write
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 247
- calib\_write.c, 820
- calib\_write\_enc, 821
- calib\_write\_bytes\_zone
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 247
- calib\_write\_config\_counter
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 248
- calib\_write\_config\_zone
  - Basic Crypto API methods for CryptoAuth Devices  
(calib\_), 248

- calib\_write\_enc
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 249
  - calib\_write.c, 821
- calib\_write\_pubkey
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 249
- calib\_write\_zone
  - Basic Crypto API methods for CryptoAuth Devices (calib\_), 249
- CAUSED
  - license.txt, 884
- cb
  - atca\_aes\_ctr\_ctx, 409
  - atca\_aes\_gcm\_ctx, 411
  - CK\_AES\_CTR\_PARAMS, 469
  - CK\_CAMELLIA\_CTR\_PARAMS, 474
- cbc\_ctx
  - atca\_aes\_cmac\_ctx, 409
- cert
  - atcacert\_build\_state\_s, 445
- cert\_def
  - atcacert\_build\_state\_s, 446
  - tng\_cert\_map\_element, 542
- cert\_elements
  - atcacert\_def\_s, 450
- cert\_elements\_count
  - atcacert\_def\_s, 450
- cert\_loc
  - atcacert\_cert\_element\_s, 447
- cert\_size
  - atcacert\_build\_state\_s, 446
- cert\_sn\_dev\_loc
  - atcacert\_def\_s, 450
- cert\_template
  - atcacert\_def\_s, 450
- cert\_template\_size
  - atcacert\_def\_s, 450
- Certificate manipulation methods (atcacert\_), 153
  - ATCA\_PACKED, 158
  - atcacert\_build\_state\_t, 162
  - atcacert\_cert\_build\_finish, 168
  - atcacert\_cert\_build\_process, 168
  - atcacert\_cert\_build\_start, 169
  - atcacert\_cert\_element\_t, 162
  - atcacert\_cert\_loc\_t, 162
  - atcacert\_cert\_sn\_src\_e, 164
  - atcacert\_cert\_sn\_src\_t, 162
  - atcacert\_cert\_type\_e, 165
  - atcacert\_cert\_type\_t, 162
  - atcacert\_create\_csr, 169
  - atcacert\_create\_csr\_pem, 170
  - atcacert\_date\_dec, 170
  - atcacert\_date\_dec\_compcert, 171
  - atcacert\_date\_dec\_iso8601\_sep, 171
  - atcacert\_date\_dec\_posix\_uint32\_be, 171
  - atcacert\_date\_dec\_posix\_uint32\_le, 172
  - atcacert\_date\_dec\_rfc5280\_gen, 172
  - atcacert\_date\_dec\_rfc5280\_utc, 172
  - atcacert\_date\_enc, 172
  - atcacert\_date\_enc\_compcert, 173
  - atcacert\_date\_enc\_iso8601\_sep, 173
  - atcacert\_date\_enc\_posix\_uint32\_be, 173
  - atcacert\_date\_enc\_posix\_uint32\_le, 173
  - atcacert\_date\_enc\_rfc5280\_gen, 173
  - atcacert\_date\_enc\_rfc5280\_utc, 174
  - atcacert\_date\_format\_e, 165
  - ATCACERT\_DATE\_FORMAT\_SIZES, 200
  - ATCACERT\_DATE\_FORMAT\_SIZES\_COUNT, 158
  - atcacert\_date\_format\_t, 163
  - atcacert\_date\_get\_max\_date, 174
  - atcacert\_def\_t, 163
  - atcacert\_der\_adjust\_length, 174
  - atcacert\_der\_dec\_ecdsa\_sig\_value, 174
  - atcacert\_der\_dec\_integer, 175
  - atcacert\_der\_dec\_length, 176
  - atcacert\_der\_enc\_ecdsa\_sig\_value, 176
  - atcacert\_der\_enc\_integer, 177
  - atcacert\_der\_enc\_length, 177
  - atcacert\_device\_loc\_t, 163
  - atcacert\_device\_zone\_e, 166
  - atcacert\_device\_zone\_t, 163
  - ATCACERT\_E\_BAD\_CERT, 158
  - ATCACERT\_E\_BAD\_PARAMS, 159
  - ATCACERT\_E\_BUFFER\_TOO\_SMALL, 159
  - ATCACERT\_E\_DECODING\_ERROR, 159
  - ATCACERT\_E\_ELEM\_MISSING, 159
  - ATCACERT\_E\_ELEM\_OUT\_OF\_BOUNDS, 159
  - ATCACERT\_E\_ERROR, 159
  - ATCACERT\_E\_INVALID\_DATE, 160
  - ATCACERT\_E\_INVALID\_TRANSFORM, 160
  - ATCACERT\_E\_SUCCESS, 160
  - ATCACERT\_E\_UNEXPECTED\_ELEM\_SIZE, 160
  - ATCACERT\_E\_UNIMPLEMENTED, 160
  - ATCACERT\_E\_VERIFY\_FAILED, 160
  - ATCACERT\_E\_WRONG\_CERT\_DEF, 161
  - atcacert\_gen\_cert\_sn, 178
  - atcacert\_gen\_challenge\_hw, 178
  - atcacert\_gen\_challenge\_sw, 179
  - atcacert\_get\_auth\_key\_id, 179
  - atcacert\_get\_cert\_element, 179
  - atcacert\_get\_cert\_sn, 180
  - atcacert\_get\_comp\_cert, 180
  - atcacert\_get\_device\_data, 181
  - atcacert\_get\_device\_locs, 181
  - atcacert\_get\_expire\_date, 182
  - atcacert\_get\_issue\_date, 183
  - atcacert\_get\_key\_id, 183
  - atcacert\_get\_response, 184
  - atcacert\_get\_signature, 184
  - atcacert\_get\_signer\_id, 185
  - atcacert\_get\_subj\_key\_id, 185
  - atcacert\_get\_subj\_public\_key, 186
  - atcacert\_get\_tbs, 186
  - atcacert\_get\_tbs\_digest, 187

- atcacert\_is\_device\_loc\_overlap, 187
- atcacert\_max\_cert\_size, 187
- atcacert\_merge\_device\_loc, 188
- atcacert\_public\_key\_add\_padding, 188
- atcacert\_public\_key\_remove\_padding, 189
- atcacert\_read\_cert, 189
- atcacert\_read\_cert\_size, 191
- atcacert\_read\_device\_loc, 191
- atcacert\_read\_subj\_key\_id, 192
- atcacert\_set\_auth\_key\_id, 192
- atcacert\_set\_auth\_key\_id\_raw, 192
- atcacert\_set\_cert\_element, 193
- atcacert\_set\_cert\_sn, 193
- atcacert\_set\_comp\_cert, 194
- atcacert\_set\_expire\_date, 195
- atcacert\_set\_issue\_date, 195
- atcacert\_set\_signature, 196
- atcacert\_set\_signer\_id, 196
- atcacert\_set\_subj\_public\_key, 197
- atcacert\_std\_cert\_element\_e, 166
- atcacert\_std\_cert\_element\_t, 163
- atcacert\_tm\_utc\_t, 163
- atcacert\_transform\_data, 197
- atcacert\_transform\_e, 166
- atcacert\_transform\_t, 163
- atcacert\_verify\_cert\_hw, 198
- atcacert\_verify\_cert\_sw, 198
- atcacert\_verify\_response\_hw, 199
- atcacert\_verify\_response\_sw, 199
- atcacert\_write\_cert, 200
- CERTTYPE\_CUSTOM, 165
- CERTTYPE\_X509, 165
- DATEFMT\_ISO8601\_SEP, 165
- DATEFMT\_ISO8601\_SEP\_SIZE, 161
- DATEFMT\_MAX\_SIZE, 161
- DATEFMT\_POSIX\_UINT32\_BE, 165
- DATEFMT\_POSIX\_UINT32\_BE\_SIZE, 161
- DATEFMT\_POSIX\_UINT32\_LE, 166
- DATEFMT\_POSIX\_UINT32\_LE\_SIZE, 161
- DATEFMT\_RFC5280\_GEN, 166
- DATEFMT\_RFC5280\_GEN\_SIZE, 161
- DATEFMT\_RFC5280\_UTC, 165
- DATEFMT\_RFC5280\_UTC\_SIZE, 161
- DEVZONE\_CONFIG, 166
- DEVZONE\_DATA, 166
- DEVZONE\_NONE, 166
- DEVZONE\_OTP, 166
- FALSE, 162
- SNSRC\_DEVICE\_SN, 165
- SNSRC\_DEVICE\_SN\_HASH, 165
- SNSRC\_DEVICE\_SN\_HASH\_POS, 165
- SNSRC\_DEVICE\_SN\_HASH\_RAW, 165
- SNSRC\_PUB\_KEY\_HASH, 165
- SNSRC\_PUB\_KEY\_HASH\_POS, 165
- SNSRC\_PUB\_KEY\_HASH\_RAW, 165
- SNSRC\_SIGNER\_ID, 165
- SNSRC\_STORED, 165
- SNSRC\_STORED\_DYNAMIC, 165
- STDCERT\_AUTH\_KEY\_ID, 166
- STDCERT\_CERT\_SN, 166
- STDCERT\_EXPIRE\_DATE, 166
- STDCERT\_ISSUE\_DATE, 166
- STDCERT\_NUM\_ELEMENTS, 166
- STDCERT\_PUBLIC\_KEY, 166
- STDCERT\_SIGNATURE, 166
- STDCERT\_SIGNER\_ID, 166
- STDCERT\_SUBJ\_KEY\_ID, 166
- TF\_BIN2HEX\_LC, 168
- TF\_BIN2HEX\_SPACE\_LC, 168
- TF\_BIN2HEX\_SPACE\_UC, 168
- TF\_BIN2HEX\_UC, 168
- TF\_HEX2BIN\_LC, 168
- TF\_HEX2BIN\_SPACE\_LC, 168
- TF\_HEX2BIN\_SPACE\_UC, 168
- TF\_HEX2BIN\_UC, 168
- TF\_NONE, 168
- TF\_REVERSE, 168
- TRUE, 162
- certificateHandle
  - CK\_CMS\_SIG\_PARAMS, 476
- CERTTYPE\_CUSTOM
  - Certificate manipulation methods (atcacert\_), 165
- CERTTYPE\_X509
  - Certificate manipulation methods (atcacert\_), 165
- cfg\_ateccx08a\_i2c\_default
  - atca\_cfgs.h, 572
- cfg\_ateccx08a\_kitcdc\_default
  - atca\_cfgs.h, 572
- cfg\_ateccx08a\_kithid\_default
  - atca\_cfgs.h, 572
- cfg\_ateccx08a\_swi\_default
  - atca\_cfgs.h, 573
- cfg\_atsha20xa\_i2c\_default
  - atca\_cfgs.h, 573
- cfg\_atsha20xa\_kitcdc\_default
  - atca\_cfgs.h, 573
- cfg\_atsha20xa\_kithid\_default
  - atca\_cfgs.h, 573
- cfg\_atsha20xa\_swi\_default
  - atca\_cfgs.h, 573
- cfg\_data
  - ATCAIfaceCfg, 461
- cfg\_zone
  - \_pkcs11\_slot\_ctx, 406
- chain\_id
  - atcacert\_def\_s, 450
- challenge
  - Host side crypto methods (atcah\_), 326
- change\_baudrate
  - i2c\_sam0\_instance, 536
  - i2c\_sam\_instance, 537
  - i2c\_start\_instance, 537
- change\_i2c\_speed
  - Hardware abstraction layer (hal\_), 270
- CHECKMAC\_CLIENT\_CHALLENGE\_IDX
  - calib\_command.h, 733

- CHECKMAC\_CLIENT\_CHALLENGE\_SIZE
  - calib\_command.h, [733](#)
- CHECKMAC\_CLIENT\_COMMAND\_SIZE
  - calib\_command.h, [733](#)
- CHECKMAC\_CLIENT\_RESPONSE\_IDX
  - calib\_command.h, [733](#)
- CHECKMAC\_CLIENT\_RESPONSE\_SIZE
  - calib\_command.h, [733](#)
- CHECKMAC\_CMD\_MATCH
  - calib\_command.h, [734](#)
- CHECKMAC\_CMD\_MISMATCH
  - calib\_command.h, [734](#)
- CHECKMAC\_COUNT
  - calib\_command.h, [734](#)
- CHECKMAC\_DATA\_IDX
  - calib\_command.h, [734](#)
- CHECKMAC\_KEYID\_IDX
  - calib\_command.h, [734](#)
- CHECKMAC\_MODE\_BLOCK1\_TEMPKEY
  - calib\_command.h, [734](#)
- CHECKMAC\_MODE\_BLOCK2\_TEMPKEY
  - calib\_command.h, [735](#)
- CHECKMAC\_MODE\_CHALLENGE
  - calib\_command.h, [735](#)
- CHECKMAC\_MODE\_IDX
  - calib\_command.h, [735](#)
- CHECKMAC\_MODE\_INCLUDE\_OTP\_64
  - calib\_command.h, [735](#)
- CHECKMAC\_MODE\_MASK
  - calib\_command.h, [735](#)
- CHECKMAC\_MODE\_SOURCE\_FLAG\_MATCH
  - calib\_command.h, [735](#)
- CHECKMAC\_OTHER\_DATA\_SIZE
  - calib\_command.h, [736](#)
- CHECKMAC\_RSP\_SIZE
  - calib\_command.h, [736](#)
- ChipMode
  - \_atecc508a\_config, [387](#)
  - \_atecc608\_config, [391](#)
  - \_atsha204a\_config, [395](#)
- ChipOptions
  - \_atecc608\_config, [391](#)
- ciphertext
  - atca\_aes\_cbc\_ctx, [407](#)
- ciphertext\_block
  - atca\_aes\_gcm\_ctx, [411](#)
- CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS, [467](#)
  - iv, [467](#)
  - length, [468](#)
  - pData, [468](#)
  - pkcs11t.h, [1038](#)
- CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR
  - pkcs11t.h, [1038](#)
- CK\_AES\_CCM\_PARAMS, [468](#)
  - pAAD, [468](#)
  - pkcs11t.h, [1039](#)
  - pNonce, [468](#)
  - ulAADLen, [468](#)
  - ulDataLen, [469](#)
  - ulMACLen, [469](#)
  - ulNonceLen, [469](#)
- CK\_AES\_CCM\_PARAMS\_PTR
  - pkcs11t.h, [1039](#)
- CK\_AES\_CTR\_PARAMS, [469](#)
  - cb, [469](#)
  - pkcs11t.h, [1039](#)
  - ulCounterBits, [469](#)
- CK\_AES\_CTR\_PARAMS\_PTR
  - pkcs11t.h, [1039](#)
- CK\_AES\_GCM\_PARAMS, [470](#)
  - pAAD, [470](#)
  - plv, [470](#)
  - pkcs11t.h, [1039](#)
  - ulAADLen, [470](#)
  - ullvBits, [470](#)
  - ullvLen, [470](#)
  - ulTagBits, [471](#)
- CK\_AES\_GCM\_PARAMS\_PTR
  - pkcs11t.h, [1039](#)
- CK\_ARIA\_CBC\_ENCRYPT\_DATA\_PARAMS, [471](#)
  - iv, [471](#)
  - length, [471](#)
  - pData, [471](#)
  - pkcs11t.h, [1039](#)
- CK\_ARIA\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR
  - pkcs11t.h, [1039](#)
- CK\_ATTRIBUTE, [471](#)
  - pkcs11t.h, [1040](#)
  - pValue, [472](#)
  - type, [472](#)
  - ulValueLen, [472](#)
- CK\_ATTRIBUTE\_PTR
  - pkcs11t.h, [1040](#)
- CK\_ATTRIBUTE\_TYPE
  - pkcs11t.h, [1040](#)
- CK\_BBOOL
  - pkcs11t.h, [1040](#)
- CK\_BYTE
  - pkcs11t.h, [1040](#)
- CK\_BYTE\_PTR
  - pkcs11t.h, [1040](#)
- CK\_C\_INITIALIZE\_ARGS, [472](#)
  - CreateMutex, [472](#)
  - DestroyMutex, [473](#)
  - flags, [473](#)
  - LockMutex, [473](#)
  - pkcs11t.h, [1040](#)
  - pReserved, [473](#)
  - UnlockMutex, [473](#)
- CK\_C\_INITIALIZE\_ARGS\_PTR
  - pkcs11t.h, [1040](#)
- CK\_CALLBACK\_FUNCTION
  - cryptoki.h, [827](#)
  - pkcs11t.h, [1062](#)
- CK\_CAMELLIA\_CBC\_ENCRYPT\_DATA\_PARAMS, [473](#)
  - iv, [474](#)



- length, 474
- pData, 474
- pkcs11t.h, 1041
- CK\_CAMELLIA\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR
  - pkcs11t.h, 1041
- CK\_CAMELLIA\_CTR\_PARAMS, 474
  - cb, 474
  - pkcs11t.h, 1041
  - ulCounterBits, 474
- CK\_CAMELLIA\_CTR\_PARAMS\_PTR
  - pkcs11t.h, 1041
- CK\_CCM\_PARAMS, 475
  - pAAD, 475
  - pkcs11t.h, 1041
  - pNonce, 475
  - ulAADLen, 475
  - ulDataLen, 475
  - ulMACLen, 475
  - ulNonceLen, 475
- CK\_CCM\_PARAMS\_PTR
  - pkcs11t.h, 1041
- CK\_CERTIFICATE\_CATEGORY
  - pkcs11t.h, 1041
- CK\_CERTIFICATE\_CATEGORY\_AUTHORITY
  - pkcs11t.h, 948
- CK\_CERTIFICATE\_CATEGORY\_OTHER\_ENTITY
  - pkcs11t.h, 948
- CK\_CERTIFICATE\_CATEGORY\_TOKEN\_USER
  - pkcs11t.h, 948
- CK\_CERTIFICATE\_CATEGORY\_UNSPECIFIED
  - pkcs11t.h, 949
- CK\_CERTIFICATE\_TYPE
  - pkcs11t.h, 1041
- CK\_CHAR
  - pkcs11t.h, 1042
- CK\_CHAR\_PTR
  - pkcs11t.h, 1042
- CK\_CMS\_SIG\_PARAMS, 476
  - certificateHandle, 476
  - pContentType, 476
  - pDigestMechanism, 476
  - pkcs11t.h, 1042
  - pRequestedAttributes, 476
  - pRequiredAttributes, 476
  - pSigningMechanism, 477
  - ulRequestedAttributesLen, 477
  - ulRequiredAttributesLen, 477
- CK\_CMS\_SIG\_PARAMS\_PTR
  - pkcs11t.h, 1042
- CK\_DATE, 477
  - day, 477
  - month, 477
  - pkcs11t.h, 1042
  - year, 478
- CK\_DECLARE\_FUNCTION
  - cryptoki.h, 827
- CK\_DECLARE\_FUNCTION\_POINTER
  - cryptoki.h, 827
- CK\_DES\_CBC\_ENCRYPT\_DATA\_PARAMS, 478
  - iv, 478
  - length, 478
  - pData, 478
  - pkcs11t.h, 1042
- CK\_DES\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR
  - pkcs11t.h, 1042
- CK\_DSA\_PARAMETER\_GEN\_PARAM, 478
  - hash, 479
  - pkcs11t.h, 1042
  - pSeed, 479
  - ulIndex, 479
  - ulSeedLen, 479
- CK\_DSA\_PARAMETER\_GEN\_PARAM\_PTR
  - pkcs11t.h, 1043
- CK\_EC\_KDF\_TYPE
  - pkcs11t.h, 1043
- CK\_ECDH1\_DERIVE\_PARAMS, 479
  - kdf, 480
  - pkcs11t.h, 1043
  - pPublicData, 480
  - pSharedData, 480
  - ulPublicDataLen, 480
  - ulSharedDataLen, 480
- CK\_ECDH1\_DERIVE\_PARAMS\_PTR
  - pkcs11t.h, 1043
- CK\_ECDH2\_DERIVE\_PARAMS, 480
  - hPrivateData, 481
  - kdf, 481
  - pkcs11t.h, 1043
  - pPublicData, 481
  - pPublicData2, 481
  - pSharedData, 481
  - ulPrivateDataLen, 481
  - ulPublicDataLen, 481
  - ulPublicDataLen2, 481
  - ulSharedDataLen, 482
- CK\_ECDH2\_DERIVE\_PARAMS\_PTR
  - pkcs11t.h, 1043
- CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS, 482
  - kdf, 482
  - pkcs11t.h, 1043
  - pSharedData, 482
  - ulAESKeyBits, 482
  - ulSharedDataLen, 482
- CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS\_PTR
  - pkcs11t.h, 1043
- CK\_ECMQV\_DERIVE\_PARAMS, 483
  - hPrivateData, 483
  - kdf, 483
  - pkcs11t.h, 1044
  - pPublicData, 483
  - pPublicData2, 483
  - pSharedData, 484
  - publicKey, 484
  - ulPrivateDataLen, 484
  - ulPublicDataLen, 484
  - ulPublicDataLen2, 484

- ulSharedDataLen, [484](#)
- CK\_ECMQV\_DERIVE\_PARAMS\_PTR
  - pkcs11t.h, [1044](#)
- CK\_EFFECTIVELY\_INFINITE
  - pkcs11t.h, [949](#)
- CK\_EXTRACT\_PARAMS
  - pkcs11t.h, [1044](#)
- CK\_EXTRACT\_PARAMS\_PTR
  - pkcs11t.h, [1044](#)
- CK\_FALSE
  - pkcs11t.h, [949](#)
- CK\_FLAGS
  - pkcs11t.h, [1044](#)
- CK\_FUNCTION\_LIST, [484](#)
  - pkcs11t.h, [1044](#)
  - version, [485](#)
- CK\_FUNCTION\_LIST\_PTR
  - pkcs11t.h, [1044](#)
- CK\_FUNCTION\_LIST\_PTR\_PTR
  - pkcs11t.h, [1044](#)
- CK\_GCM\_PARAMS, [485](#)
  - pAAD, [485](#)
  - plv, [485](#)
  - pkcs11t.h, [1045](#)
  - ulAADLen, [485](#)
  - ullvBits, [486](#)
  - ullvLen, [486](#)
  - ulTagBits, [486](#)
- CK\_GCM\_PARAMS\_PTR
  - pkcs11t.h, [1045](#)
- CK\_GOSTR3410\_DERIVE\_PARAMS, [486](#)
  - kdf, [486](#)
  - pkcs11t.h, [1045](#)
  - pPublicData, [486](#)
  - pUKM, [487](#)
  - ulPublicDataLen, [487](#)
  - ulUKMLen, [487](#)
- CK\_GOSTR3410\_DERIVE\_PARAMS\_PTR
  - pkcs11t.h, [1045](#)
- CK\_GOSTR3410\_KEY\_WRAP\_PARAMS, [487](#)
  - hKey, [487](#)
  - pkcs11t.h, [1045](#)
  - pUKM, [487](#)
  - pWrapOID, [488](#)
  - ulUKMLen, [488](#)
  - ulWrapOIDLen, [488](#)
- CK\_GOSTR3410\_KEY\_WRAP\_PARAMS\_PTR
  - pkcs11t.h, [1045](#)
- CK\_HW\_FEATURE\_TYPE
  - pkcs11t.h, [1045](#)
- CK\_INFO, [488](#)
  - cryptokiVersion, [488](#)
  - flags, [488](#)
  - libraryDescription, [489](#)
  - libraryVersion, [489](#)
  - manufacturerID, [489](#)
  - pkcs11t.h, [1045](#)
- CK\_INFO\_PTR
  - pkcs11t.h, [1046](#)
- CK\_INVALID\_HANDLE
  - pkcs11t.h, [949](#)
- CK\_JAVA\_MIDP\_SECURITY\_DOMAIN
  - pkcs11t.h, [1046](#)
- CK\_KEA\_DERIVE\_PARAMS, [489](#)
  - isSender, [489](#)
  - pkcs11t.h, [1046](#)
  - pPublicData, [489](#)
  - pRandomA, [490](#)
  - pRandomB, [490](#)
  - ulPublicDataLen, [490](#)
  - ulRandomLen, [490](#)
- CK\_KEA\_DERIVE\_PARAMS\_PTR
  - pkcs11t.h, [1046](#)
- CK\_KEY\_DERIVATION\_STRING\_DATA, [490](#)
  - pData, [490](#)
  - pkcs11t.h, [1046](#)
  - ulLen, [491](#)
- CK\_KEY\_DERIVATION\_STRING\_DATA\_PTR
  - pkcs11t.h, [1046](#)
- CK\_KEY\_TYPE
  - pkcs11t.h, [1046](#)
- CK\_KEY\_WRAP\_SET\_OAEP\_PARAMS, [491](#)
  - bBC, [491](#)
  - pkcs11t.h, [1046](#)
  - pX, [491](#)
  - ulXLen, [491](#)
- CK\_KEY\_WRAP\_SET\_OAEP\_PARAMS\_PTR
  - pkcs11t.h, [1047](#)
- CK\_KIP\_PARAMS, [491](#)
  - hKey, [492](#)
  - pkcs11t.h, [1047](#)
  - pMechanism, [492](#)
  - pSeed, [492](#)
  - ulSeedLen, [492](#)
- CK\_KIP\_PARAMS\_PTR
  - pkcs11t.h, [1047](#)
- CK\_LONG
  - pkcs11t.h, [1047](#)
- CK\_MAC\_GENERAL\_PARAMS
  - pkcs11t.h, [1047](#)
- CK\_MAC\_GENERAL\_PARAMS\_PTR
  - pkcs11t.h, [1047](#)
- CK\_MECHANISM, [492](#)
  - mechanism, [493](#)
  - pkcs11t.h, [1047](#)
  - pParameter, [493](#)
  - ulParameterLen, [493](#)
- CK\_MECHANISM\_INFO, [493](#)
  - flags, [493](#)
  - pkcs11t.h, [1047](#)
  - ulMaxKeySize, [493](#)
  - ulMinKeySize, [494](#)
- CK\_MECHANISM\_INFO\_PTR
  - pkcs11t.h, [1048](#)
- CK\_MECHANISM\_PTR
  - pkcs11t.h, [1048](#)



---

CK\_MECHANISM\_TYPE  
     pkcs11t.h, [1048](#)  
 CK\_MECHANISM\_TYPE\_PTR  
     pkcs11t.h, [1048](#)  
 CK\_NEED\_ARG\_LIST  
     pkcs11.h, [895](#)  
 CK\_NOTIFICATION  
     pkcs11t.h, [1048](#)  
 CK\_OBJECT\_CLASS  
     pkcs11t.h, [1048](#)  
 CK\_OBJECT\_CLASS\_PTR  
     pkcs11t.h, [1048](#)  
 CK\_OBJECT\_HANDLE  
     pkcs11t.h, [1048](#)  
 CK\_OBJECT\_HANDLE\_PTR  
     pkcs11t.h, [1049](#)  
 CK\_OTP\_CHALLENGE  
     pkcs11t.h, [949](#)  
 CK\_OTP\_COUNTER  
     pkcs11t.h, [949](#)  
 CK\_OTP\_FLAGS  
     pkcs11t.h, [949](#)  
 CK\_OTP\_FORMAT\_ALPHANUMERIC  
     pkcs11t.h, [949](#)  
 CK\_OTP\_FORMAT\_BINARY  
     pkcs11t.h, [950](#)  
 CK\_OTP\_FORMAT\_DECIMAL  
     pkcs11t.h, [950](#)  
 CK\_OTP\_FORMAT\_HEXADecimal  
     pkcs11t.h, [950](#)  
 CK\_OTP\_OUTPUT\_FORMAT  
     pkcs11t.h, [950](#)  
 CK\_OTP\_OUTPUT\_LENGTH  
     pkcs11t.h, [950](#)  
 CK\_OTP\_PARAM, [494](#)  
     pkcs11t.h, [1049](#)  
     pValue, [494](#)  
     type, [494](#)  
     ulValueLen, [494](#)  
 CK\_OTP\_PARAM\_IGNORED  
     pkcs11t.h, [950](#)  
 CK\_OTP\_PARAM\_MANDATORY  
     pkcs11t.h, [950](#)  
 CK\_OTP\_PARAM\_OPTIONAL  
     pkcs11t.h, [950](#)  
 CK\_OTP\_PARAM\_PTR  
     pkcs11t.h, [1049](#)  
 CK\_OTP\_PARAM\_TYPE  
     pkcs11t.h, [1049](#)  
 CK\_OTP\_PARAMS, [494](#)  
     pkcs11t.h, [1049](#)  
     pParams, [495](#)  
     ulCount, [495](#)  
 CK\_OTP\_PARAMS\_PTR  
     pkcs11t.h, [1049](#)  
 CK\_OTP\_PIN  
     pkcs11t.h, [951](#)  
 CK\_OTP\_SIGNATURE\_INFO, [495](#)  
     pkcs11t.h, [1049](#)  
     pParams, [495](#)  
     ulCount, [495](#)  
 CK\_OTP\_SIGNATURE\_INFO\_PTR  
     pkcs11t.h, [1049](#)  
 CK\_OTP\_TIME  
     pkcs11t.h, [951](#)  
 CK\_OTP\_VALUE  
     pkcs11t.h, [951](#)  
 CK\_PARAM\_TYPE  
     pkcs11t.h, [1050](#)  
 CK\_PBE\_PARAMS, [496](#)  
     pInitVector, [496](#)  
     pkcs11t.h, [1050](#)  
     pPassword, [496](#)  
     pSalt, [496](#)  
     ulIteration, [496](#)  
     ulPasswordLen, [496](#)  
     ulSaltLen, [496](#)  
 CK\_PBE\_PARAMS\_PTR  
     pkcs11t.h, [1050](#)  
 CK\_PKCS11\_FUNCTION\_INFO  
     pkcs11.h, [895](#)  
 CK\_PKCS5\_PBKD2\_PARAMS, [497](#)  
     iterations, [497](#)  
     pkcs11t.h, [1050](#)  
     pPassword, [497](#)  
     pPrfData, [497](#)  
     prf, [497](#)  
     pSaltSourceData, [498](#)  
     saltSource, [498](#)  
     ulPasswordLen, [498](#)  
     ulPrfDataLen, [498](#)  
     ulSaltSourceDataLen, [498](#)  
 CK\_PKCS5\_PBKD2\_PARAMS2, [498](#)  
     iterations, [499](#)  
     pkcs11t.h, [1050](#)  
     pPassword, [499](#)  
     pPrfData, [499](#)  
     prf, [499](#)  
     pSaltSourceData, [499](#)  
     saltSource, [499](#)  
     ulPasswordLen, [499](#)  
     ulPrfDataLen, [499](#)  
     ulSaltSourceDataLen, [500](#)  
 CK\_PKCS5\_PBKD2\_PARAMS2\_PTR  
     pkcs11t.h, [1050](#)  
 CK\_PKCS5\_PBKD2\_PARAMS\_PTR  
     pkcs11t.h, [1050](#)  
 CK\_PKCS5\_PBKD2\_PSEUDO\_RANDOM\_FUNCTION\_TYPE  
     pkcs11t.h, [1050](#)  
 CK\_PKCS5\_PBKD2\_PSEUDO\_RANDOM\_FUNCTION\_TYPE\_PTR  
     pkcs11t.h, [1051](#)  
 CK\_PKCS5\_PBKDF2\_SALT\_SOURCE\_TYPE  
     pkcs11t.h, [1051](#)  
 CK\_PKCS5\_PBKDF2\_SALT\_SOURCE\_TYPE\_PTR  
     pkcs11t.h, [1051](#)  
 CK\_PTR

- cryptoki.h, [827](#)
- CK\_RC2\_CBC\_PARAMS, [500](#)
  - iv, [500](#)
  - pkcs11t.h, [1051](#)
  - ulEffectiveBits, [500](#)
- CK\_RC2\_CBC\_PARAMS\_PTR
  - pkcs11t.h, [1051](#)
- CK\_RC2\_MAC\_GENERAL\_PARAMS, [500](#)
  - pkcs11t.h, [1051](#)
  - ulEffectiveBits, [501](#)
  - ulMacLength, [501](#)
- CK\_RC2\_MAC\_GENERAL\_PARAMS\_PTR
  - pkcs11t.h, [1051](#)
- CK\_RC2\_PARAMS
  - pkcs11t.h, [1051](#)
- CK\_RC2\_PARAMS\_PTR
  - pkcs11t.h, [1052](#)
- CK\_RC5\_CBC\_PARAMS, [501](#)
  - plv, [501](#)
  - pkcs11t.h, [1052](#)
  - ullvLen, [501](#)
  - ulRounds, [501](#)
  - ulWordsize, [502](#)
- CK\_RC5\_CBC\_PARAMS\_PTR
  - pkcs11t.h, [1052](#)
- CK\_RC5\_MAC\_GENERAL\_PARAMS, [502](#)
  - pkcs11t.h, [1052](#)
  - ulMacLength, [502](#)
  - ulRounds, [502](#)
  - ulWordsize, [502](#)
- CK\_RC5\_MAC\_GENERAL\_PARAMS\_PTR
  - pkcs11t.h, [1052](#)
- CK\_RC5\_PARAMS, [502](#)
  - pkcs11t.h, [1052](#)
  - ulRounds, [503](#)
  - ulWordsize, [503](#)
- CK\_RC5\_PARAMS\_PTR
  - pkcs11t.h, [1052](#)
- CK\_RSA\_AES\_KEY\_WRAP\_PARAMS, [503](#)
  - pkcs11t.h, [1052](#)
  - pOAEPParams, [503](#)
  - ulAESKeyBits, [503](#)
- CK\_RSA\_AES\_KEY\_WRAP\_PARAMS\_PTR
  - pkcs11t.h, [1053](#)
- CK\_RSA\_PKCS\_MGF\_TYPE
  - pkcs11t.h, [1053](#)
- CK\_RSA\_PKCS\_MGF\_TYPE\_PTR
  - pkcs11t.h, [1053](#)
- CK\_RSA\_PKCS\_OAEP\_PARAMS, [504](#)
  - hashAlg, [504](#)
  - mgf, [504](#)
  - pkcs11t.h, [1053](#)
  - pSourceData, [504](#)
  - source, [504](#)
  - ulSourceDataLen, [504](#)
- CK\_RSA\_PKCS\_OAEP\_PARAMS\_PTR
  - pkcs11t.h, [1053](#)
- CK\_RSA\_PKCS\_OAEP\_SOURCE\_TYPE
  - pkcs11t.h, [1053](#)
- CK\_RSA\_PKCS\_OAEP\_SOURCE\_TYPE\_PTR
  - pkcs11t.h, [1053](#)
- CK\_RSA\_PKCS\_PSS\_PARAMS, [505](#)
  - hashAlg, [505](#)
  - mgf, [505](#)
  - pkcs11t.h, [1053](#)
  - sLen, [505](#)
- CK\_RSA\_PKCS\_PSS\_PARAMS\_PTR
  - pkcs11t.h, [1054](#)
- CK\_RV
  - pkcs11t.h, [1054](#)
- CK\_SECURITY\_DOMAIN\_MANUFACTURER
  - pkcs11t.h, [951](#)
- CK\_SECURITY\_DOMAIN\_OPERATOR
  - pkcs11t.h, [951](#)
- CK\_SECURITY\_DOMAIN\_THIRD\_PARTY
  - pkcs11t.h, [951](#)
- CK\_SECURITY\_DOMAIN\_UNSPECIFIED
  - pkcs11t.h, [951](#)
- CK\_SEED\_CBC\_ENCRYPT\_DATA\_PARAMS, [505](#)
  - iv, [505](#)
  - length, [506](#)
  - pData, [506](#)
  - pkcs11t.h, [1054](#)
- CK\_SEED\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR
  - pkcs11t.h, [1054](#)
- CK\_SESSION\_HANDLE
  - pkcs11t.h, [1054](#)
- CK\_SESSION\_HANDLE\_PTR
  - pkcs11t.h, [1054](#)
- CK\_SESSION\_INFO, [506](#)
  - flags, [506](#)
  - pkcs11t.h, [1054](#)
  - slotID, [506](#)
  - state, [506](#)
  - ulDeviceError, [507](#)
- CK\_SESSION\_INFO\_PTR
  - pkcs11t.h, [1054](#)
- CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS, [507](#)
  - pBaseG, [507](#)
  - pkcs11t.h, [1055](#)
  - pPassword, [507](#)
  - pPrimeP, [507](#)
  - pPublicData, [508](#)
  - pRandomA, [508](#)
  - pSubprimeQ, [508](#)
  - ulPAndGLen, [508](#)
  - ulPasswordLen, [508](#)
  - ulPublicDataLen, [508](#)
  - ulQLen, [508](#)
  - ulRandomLen, [508](#)
- CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS\_PTR
  - pkcs11t.h, [1055](#)
- CK\_SKIPJACK\_RELAYX\_PARAMS, [509](#)
  - pkcs11t.h, [1055](#)
  - pNewPassword, [509](#)
  - pNewPublicData, [509](#)

- pNewRandomA, 509
- pOldPassword, 509
- pOldPublicData, 510
- pOldRandomA, 510
- pOldWrappedX, 510
- ulNewPasswordLen, 510
- ulNewPublicDataLen, 510
- ulNewRandomLen, 510
- ulOldPasswordLen, 510
- ulOldPublicDataLen, 510
- ulOldRandomLen, 511
- ulOldWrappedXLen, 511
- CK\_SKIPJACK\_RELAYX\_PARAMS\_PTR
  - pkcs11t.h, 1055
- CK\_SLOT\_ID
  - pkcs11t.h, 1055
- CK\_SLOT\_ID\_PTR
  - pkcs11t.h, 1055
- CK\_SLOT\_INFO, 511
  - firmwareVersion, 511
  - flags, 511
  - hardwareVersion, 511
  - manufacturerID, 512
  - pkcs11t.h, 1055
  - slotDescription, 512
- CK\_SLOT\_INFO\_PTR
  - pkcs11t.h, 1055
- CK\_SSL3\_KEY\_MAT\_OUT, 512
  - hClientKey, 512
  - hClientMacSecret, 512
  - hServerKey, 512
  - hServerMacSecret, 513
  - pIVClient, 513
  - pIVServer, 513
  - pkcs11t.h, 1056
- CK\_SSL3\_KEY\_MAT\_OUT\_PTR
  - pkcs11t.h, 1056
- CK\_SSL3\_KEY\_MAT\_PARAMS, 513
  - blsExport, 513
  - pkcs11t.h, 1056
  - pReturnedKeyMaterial, 513
  - RandomInfo, 514
  - ulIVSizeInBits, 514
  - ulKeySizeInBits, 514
  - ulMacSizeInBits, 514
- CK\_SSL3\_KEY\_MAT\_PARAMS\_PTR
  - pkcs11t.h, 1056
- CK\_SSL3\_MASTER\_KEY\_DERIVE\_PARAMS, 514
  - pkcs11t.h, 1056
  - pVersion, 514
  - RandomInfo, 515
- CK\_SSL3\_MASTER\_KEY\_DERIVE\_PARAMS\_PTR
  - pkcs11t.h, 1056
- CK\_SSL3\_RANDOM\_DATA, 515
  - pClientRandom, 515
  - pkcs11t.h, 1056
  - pServerRandom, 515
  - ulClientRandomLen, 515
  - ulServerRandomLen, 515
- CK\_STATE
  - pkcs11t.h, 1056
- CK\_TLS12\_KEY\_MAT\_PARAMS, 516
  - blsExport, 516
  - pkcs11t.h, 1057
  - pReturnedKeyMaterial, 516
  - prfHashMechanism, 516
  - RandomInfo, 516
  - ulIVSizeInBits, 516
  - ulKeySizeInBits, 517
  - ulMacSizeInBits, 517
- CK\_TLS12\_KEY\_MAT\_PARAMS\_PTR
  - pkcs11t.h, 1057
- CK\_TLS12\_MASTER\_KEY\_DERIVE\_PARAMS, 517
  - pkcs11t.h, 1057
  - prfHashMechanism, 517
  - pVersion, 517
  - RandomInfo, 517
- CK\_TLS12\_MASTER\_KEY\_DERIVE\_PARAMS\_PTR
  - pkcs11t.h, 1057
- CK\_TLS\_KDF\_PARAMS, 518
  - pContextData, 518
  - pkcs11t.h, 1057
  - pLabel, 518
  - prfMechanism, 518
  - RandomInfo, 518
  - ulContextDataLength, 518
  - ulLabelLength, 519
- CK\_TLS\_KDF\_PARAMS\_PTR
  - pkcs11t.h, 1057
- CK\_TLS\_MAC\_PARAMS, 519
  - pkcs11t.h, 1057
  - prfHashMechanism, 519
  - ulMacLength, 519
  - ulServerOrClient, 519
- CK\_TLS\_MAC\_PARAMS\_PTR
  - pkcs11t.h, 1057
- CK\_TLS\_PRF\_PARAMS, 519
  - pkcs11t.h, 1058
  - pLabel, 520
  - pOutput, 520
  - pSeed, 520
  - pulOutputLen, 520
  - ulLabelLen, 520
  - ulSeedLen, 520
- CK\_TLS\_PRF\_PARAMS\_PTR
  - pkcs11t.h, 1058
- CK\_TOKEN\_INFO, 521
  - firmwareVersion, 521
  - flags, 521
  - hardwareVersion, 521
  - label, 521
  - manufacturerID, 522
  - model, 522
  - pkcs11t.h, 1058
  - serialNumber, 522
  - ulFreePrivateMemory, 522

- ulFreePublicMemory, [522](#)
- ulMaxPinLen, [522](#)
- ulMaxRwSessionCount, [522](#)
- ulMaxSessionCount, [522](#)
- ulMinPinLen, [523](#)
- ulRwSessionCount, [523](#)
- ulSessionCount, [523](#)
- ulTotalPrivateMemory, [523](#)
- ulTotalPublicMemory, [523](#)
- utcTime, [523](#)
- CK\_TOKEN\_INFO\_PTR
  - pkcs11t.h, [1058](#)
- CK\_TRUE
  - pkcs11t.h, [951](#)
- CK\_ULONG
  - pkcs11t.h, [1058](#)
- CK\_ULONG\_PTR
  - pkcs11t.h, [1058](#)
- CK\_UNAVAILABLE\_INFORMATION
  - pkcs11t.h, [952](#)
- CK\_USER\_TYPE
  - pkcs11t.h, [1058](#)
- CK\_UTF8CHAR
  - pkcs11t.h, [1058](#)
- CK\_UTF8CHAR\_PTR
  - pkcs11t.h, [1059](#)
- CK\_VERSION, [523](#)
  - major, [524](#)
  - minor, [524](#)
  - pkcs11t.h, [1059](#)
- CK\_VERSION\_PTR
  - pkcs11t.h, [1059](#)
- CK\_VOID\_PTR
  - pkcs11t.h, [1059](#)
- CK\_VOID\_PTR\_PTR
  - pkcs11t.h, [1059](#)
- CK\_WTLS\_KEY\_MAT\_OUT, [524](#)
  - hKey, [524](#)
  - hMacSecret, [524](#)
  - pIV, [524](#)
  - pkcs11t.h, [1059](#)
- CK\_WTLS\_KEY\_MAT\_OUT\_PTR
  - pkcs11t.h, [1059](#)
- CK\_WTLS\_KEY\_MAT\_PARAMS, [525](#)
  - blsExport, [525](#)
  - DigestMechanism, [525](#)
  - pkcs11t.h, [1059](#)
  - pReturnedKeyMaterial, [525](#)
  - RandomInfo, [525](#)
  - ulIVSizeInBits, [525](#)
  - ulKeySizeInBits, [526](#)
  - ulMacSizeInBits, [526](#)
  - ulSequenceNumber, [526](#)
- CK\_WTLS\_KEY\_MAT\_PARAMS\_PTR
  - pkcs11t.h, [1060](#)
- CK\_WTLS\_MASTER\_KEY\_DERIVE\_PARAMS, [526](#)
  - DigestMechanism, [526](#)
  - pkcs11t.h, [1060](#)
  - pVersion, [526](#)
  - RandomInfo, [527](#)
- CK\_WTLS\_MASTER\_KEY\_DERIVE\_PARAMS\_PTR
  - pkcs11t.h, [1060](#)
- CK\_WTLS\_PRF\_PARAMS, [527](#)
  - DigestMechanism, [527](#)
  - pkcs11t.h, [1060](#)
  - pLabel, [527](#)
  - pOutput, [527](#)
  - pSeed, [527](#)
  - pulOutputLen, [528](#)
  - ulLabelLen, [528](#)
  - ulSeedLen, [528](#)
- CK\_WTLS\_PRF\_PARAMS\_PTR
  - pkcs11t.h, [1060](#)
- CK\_WTLS\_RANDOM\_DATA, [528](#)
  - pClientRandom, [528](#)
  - pkcs11t.h, [1060](#)
  - pServerRandom, [528](#)
  - ulClientRandomLen, [529](#)
  - ulServerRandomLen, [529](#)
- CK\_WTLS\_RANDOM\_DATA\_PTR
  - pkcs11t.h, [1060](#)
- CK\_X9\_42\_DH1\_DERIVE\_PARAMS, [529](#)
  - kdf, [529](#)
  - pkcs11t.h, [1060](#)
  - pOtherInfo, [529](#)
  - pPublicData, [529](#)
  - ulOtherInfoLen, [530](#)
  - ulPublicDataLen, [530](#)
- CK\_X9\_42\_DH1\_DERIVE\_PARAMS\_PTR
  - pkcs11t.h, [1061](#)
- CK\_X9\_42\_DH2\_DERIVE\_PARAMS, [530](#)
  - hPrivateData, [530](#)
  - kdf, [530](#)
  - pkcs11t.h, [1061](#)
  - pOtherInfo, [531](#)
  - pPublicData, [531](#)
  - pPublicData2, [531](#)
  - ulOtherInfoLen, [531](#)
  - ulPrivateDataLen, [531](#)
  - ulPublicDataLen, [531](#)
  - ulPublicDataLen2, [531](#)
- CK\_X9\_42\_DH2\_DERIVE\_PARAMS\_PTR
  - pkcs11t.h, [1061](#)
- CK\_X9\_42\_DH\_KDF\_TYPE
  - pkcs11t.h, [1061](#)
- CK\_X9\_42\_DH\_KDF\_TYPE\_PTR
  - pkcs11t.h, [1061](#)
- CK\_X9\_42\_MQV\_DERIVE\_PARAMS, [532](#)
  - hPrivateData, [532](#)
  - kdf, [532](#)
  - pkcs11t.h, [1061](#)
  - pOtherInfo, [532](#)
  - pPublicData, [532](#)
  - pPublicData2, [532](#)
  - publicKey, [533](#)
  - ulOtherInfoLen, [533](#)

ulPrivateDataLen, [533](#)  
ulPublicDataLen, [533](#)  
ulPublicDataLen2, [533](#)  
CK\_X9\_42\_MQV\_DERIVE\_PARAMS\_PTR  
    pkcs11t.h, [1061](#)  
CKA\_AC\_ISSUER  
    pkcs11t.h, [952](#)  
CKA\_ALLOWED\_MECHANISMS  
    pkcs11t.h, [952](#)  
CKA\_ALWAYS\_AUTHENTICATE  
    pkcs11t.h, [952](#)  
CKA\_ALWAYS\_SENSITIVE  
    pkcs11t.h, [952](#)  
CKA\_APPLICATION  
    pkcs11t.h, [952](#)  
CKA\_ATTR\_TYPES  
    pkcs11t.h, [952](#)  
CKA\_AUTH\_PIN\_FLAGS  
    pkcs11t.h, [952](#)  
CKA\_BASE  
    pkcs11t.h, [953](#)  
CKA\_BITS\_PER\_PIXEL  
    pkcs11t.h, [953](#)  
CKA\_CERTIFICATE\_CATEGORY  
    pkcs11t.h, [953](#)  
CKA\_CERTIFICATE\_TYPE  
    pkcs11t.h, [953](#)  
CKA\_CHAR\_COLUMNS  
    pkcs11t.h, [953](#)  
CKA\_CHAR\_ROWS  
    pkcs11t.h, [953](#)  
CKA\_CHAR\_SETS  
    pkcs11t.h, [953](#)  
CKA\_CHECK\_VALUE  
    pkcs11t.h, [953](#)  
CKA\_CLASS  
    pkcs11t.h, [954](#)  
CKA\_COEFFICIENT  
    pkcs11t.h, [954](#)  
CKA\_COLOR  
    pkcs11t.h, [954](#)  
CKA\_COPYABLE  
    pkcs11t.h, [954](#)  
CKA\_DECRYPT  
    pkcs11t.h, [954](#)  
CKA\_DEFAULT\_CMS\_ATTRIBUTES  
    pkcs11t.h, [954](#)  
CKA\_DERIVE  
    pkcs11t.h, [954](#)  
CKA\_DERIVE\_TEMPLATE  
    pkcs11t.h, [954](#)  
CKA\_DESTROYABLE  
    pkcs11t.h, [955](#)  
CKA\_EC\_PARAMS  
    pkcs11t.h, [955](#)  
CKA\_EC\_POINT  
    pkcs11t.h, [955](#)  
CKA\_ECDSA\_PARAMS  
    pkcs11t.h, [955](#)  
CKA\_ENCODING\_METHODS  
    pkcs11t.h, [955](#)  
CKA\_ENCRYPT  
    pkcs11t.h, [955](#)  
CKA\_END\_DATE  
    pkcs11t.h, [955](#)  
CKA\_EXPONENT\_1  
    pkcs11t.h, [955](#)  
CKA\_EXPONENT\_2  
    pkcs11t.h, [956](#)  
CKA\_EXTRACTABLE  
    pkcs11t.h, [956](#)  
CKA\_GOST28147\_PARAMS  
    pkcs11t.h, [956](#)  
CKA\_GOSTR3410\_PARAMS  
    pkcs11t.h, [956](#)  
CKA\_GOSTR3411\_PARAMS  
    pkcs11t.h, [956](#)  
CKA\_HAS\_RESET  
    pkcs11t.h, [956](#)  
CKA\_HASH\_OF\_ISSUER\_PUBLIC\_KEY  
    pkcs11t.h, [956](#)  
CKA\_HASH\_OF\_SUBJECT\_PUBLIC\_KEY  
    pkcs11t.h, [956](#)  
CKA\_HW\_FEATURE\_TYPE  
    pkcs11t.h, [957](#)  
CKA\_ID  
    pkcs11t.h, [957](#)  
CKA\_ISSUER  
    pkcs11t.h, [957](#)  
CKA\_JAVA\_MIDP\_SECURITY\_DOMAIN  
    pkcs11t.h, [957](#)  
CKA\_KEY\_GEN\_MECHANISM  
    pkcs11t.h, [957](#)  
CKA\_KEY\_TYPE  
    pkcs11t.h, [957](#)  
CKA\_LABEL  
    pkcs11t.h, [957](#)  
CKA\_LOCAL  
    pkcs11t.h, [957](#)  
CKA\_MECHANISM\_TYPE  
    pkcs11t.h, [958](#)  
CKA\_MIME\_TYPES  
    pkcs11t.h, [958](#)  
CKA\_MODIFIABLE  
    pkcs11t.h, [958](#)  
CKA\_MODULUS  
    pkcs11t.h, [958](#)  
CKA\_MODULUS\_BITS  
    pkcs11t.h, [958](#)  
CKA\_NAME\_HASH\_ALGORITHM  
    pkcs11t.h, [958](#)  
CKA\_NEVER\_EXTRACTABLE  
    pkcs11t.h, [958](#)  
CKA\_OBJECT\_ID  
    pkcs11t.h, [958](#)  
CKA\_OTP\_CHALLENGE\_REQUIREMENT

[pkcs11t.h, 959](#)  
 CKA\_OTP\_COUNTER  
[pkcs11t.h, 959](#)  
 CKA\_OTP\_COUNTER\_REQUIREMENT  
[pkcs11t.h, 959](#)  
 CKA\_OTP\_FORMAT  
[pkcs11t.h, 959](#)  
 CKA\_OTP\_LENGTH  
[pkcs11t.h, 959](#)  
 CKA\_OTP\_PIN\_REQUIREMENT  
[pkcs11t.h, 959](#)  
 CKA\_OTP\_SERVICE\_IDENTIFIER  
[pkcs11t.h, 959](#)  
 CKA\_OTP\_SERVICE\_LOGO  
[pkcs11t.h, 959](#)  
 CKA\_OTP\_SERVICE\_LOGO\_TYPE  
[pkcs11t.h, 960](#)  
 CKA\_OTP\_TIME  
[pkcs11t.h, 960](#)  
 CKA\_OTP\_TIME\_INTERVAL  
[pkcs11t.h, 960](#)  
 CKA\_OTP\_TIME\_REQUIREMENT  
[pkcs11t.h, 960](#)  
 CKA\_OTP\_USER\_FRIENDLY\_MODE  
[pkcs11t.h, 960](#)  
 CKA\_OTP\_USER\_IDENTIFIER  
[pkcs11t.h, 960](#)  
 CKA\_OWNER  
[pkcs11t.h, 960](#)  
 CKA\_PIXEL\_X  
[pkcs11t.h, 960](#)  
 CKA\_PIXEL\_Y  
[pkcs11t.h, 961](#)  
 CKA\_PRIME  
[pkcs11t.h, 961](#)  
 CKA\_PRIME\_1  
[pkcs11t.h, 961](#)  
 CKA\_PRIME\_2  
[pkcs11t.h, 961](#)  
 CKA\_PRIME\_BITS  
[pkcs11t.h, 961](#)  
 CKA\_PRIVATE  
[pkcs11t.h, 961](#)  
 CKA\_PRIVATE\_EXPONENT  
[pkcs11t.h, 961](#)  
 CKA\_PUBLIC\_EXPONENT  
[pkcs11t.h, 961](#)  
 CKA\_PUBLIC\_KEY\_INFO  
[pkcs11t.h, 962](#)  
 CKA\_REQUIRED\_CMS\_ATTRIBUTES  
[pkcs11t.h, 962](#)  
 CKA\_RESET\_ON\_INIT  
[pkcs11t.h, 962](#)  
 CKA\_RESOLUTION  
[pkcs11t.h, 962](#)  
 CKA\_SECONDARY\_AUTH  
[pkcs11t.h, 962](#)  
 CKA\_SENSITIVE  
[pkcs11t.h, 962](#)  
 CKA\_SERIAL\_NUMBER  
[pkcs11t.h, 962](#)  
 CKA\_SIGN  
[pkcs11t.h, 962](#)  
 CKA\_SIGN\_RECOVER  
[pkcs11t.h, 963](#)  
 CKA\_START\_DATE  
[pkcs11t.h, 963](#)  
 CKA\_SUB\_PRIME\_BITS  
[pkcs11t.h, 963](#)  
 CKA\_SUBJECT  
[pkcs11t.h, 963](#)  
 CKA\_SUBPRIME  
[pkcs11t.h, 963](#)  
 CKA\_SUBPRIME\_BITS  
[pkcs11t.h, 963](#)  
 CKA\_SUPPORTED\_CMS\_ATTRIBUTES  
[pkcs11t.h, 963](#)  
 CKA\_TOKEN  
[pkcs11t.h, 963](#)  
 CKA\_TRUSTED  
[pkcs11t.h, 964](#)  
 CKA\_UNWRAP  
[pkcs11t.h, 964](#)  
 CKA\_UNWRAP\_TEMPLATE  
[pkcs11t.h, 964](#)  
 CKA\_URL  
[pkcs11t.h, 964](#)  
 CKA\_VALUE  
[pkcs11t.h, 964](#)  
 CKA\_VALUE\_BITS  
[pkcs11t.h, 964](#)  
 CKA\_VALUE\_LEN  
[pkcs11t.h, 964](#)  
 CKA\_VENDOR\_DEFINED  
[pkcs11t.h, 964](#)  
 CKA\_VERIFY  
[pkcs11t.h, 965](#)  
 CKA\_VERIFY\_RECOVER  
[pkcs11t.h, 965](#)  
 CKA\_WRAP  
[pkcs11t.h, 965](#)  
 CKA\_WRAP\_TEMPLATE  
[pkcs11t.h, 965](#)  
 CKA\_WRAP\_WITH\_TRUSTED  
[pkcs11t.h, 965](#)  
 CKC\_OPENPGP  
[pkcs11t.h, 965](#)  
 CKC\_VENDOR\_DEFINED  
[pkcs11t.h, 965](#)  
 CKC\_WTLS  
[pkcs11t.h, 965](#)  
 CKC\_X\_509  
[pkcs11t.h, 966](#)  
 CKC\_X\_509\_ATTR\_CERT  
[pkcs11t.h, 966](#)  
 CKD\_CPDIVERSIFY\_KDF

pkcs11t.h, [966](#)  
CKD\_NULL  
pkcs11t.h, [966](#)  
CKD\_SHA1\_KDF  
pkcs11t.h, [966](#)  
CKD\_SHA1\_KDF\_ASN1  
pkcs11t.h, [966](#)  
CKD\_SHA1\_KDF\_CONCATENATE  
pkcs11t.h, [966](#)  
CKD\_SHA224\_KDF  
pkcs11t.h, [966](#)  
CKD\_SHA256\_KDF  
pkcs11t.h, [967](#)  
CKD\_SHA384\_KDF  
pkcs11t.h, [967](#)  
CKD\_SHA512\_KDF  
pkcs11t.h, [967](#)  
CKF\_ARRAY\_ATTRIBUTE  
pkcs11t.h, [967](#)  
CKF\_CLOCK\_ON\_TOKEN  
pkcs11t.h, [967](#)  
CKF\_DECRYPT  
pkcs11t.h, [967](#)  
CKF\_DERIVE  
pkcs11t.h, [967](#)  
CKF\_DIGEST  
pkcs11t.h, [967](#)  
CKF\_DONT\_BLOCK  
pkcs11t.h, [968](#)  
CKF\_DUAL\_CRYPT\_OPERATIONS  
pkcs11t.h, [968](#)  
CKF\_EC\_COMPRESS  
pkcs11t.h, [968](#)  
CKF\_EC\_ECPARAMETERS  
pkcs11t.h, [968](#)  
CKF\_EC\_F\_2M  
pkcs11t.h, [968](#)  
CKF\_EC\_F\_P  
pkcs11t.h, [968](#)  
CKF\_EC\_NAMEDCURVE  
pkcs11t.h, [968](#)  
CKF\_EC\_UNCOMPRESS  
pkcs11t.h, [968](#)  
CKF\_ENCRYPT  
pkcs11t.h, [969](#)  
CKF\_ERROR\_STATE  
pkcs11t.h, [969](#)  
CKF\_EXCLUDE\_CHALLENGE  
pkcs11t.h, [969](#)  
CKF\_EXCLUDE\_COUNTER  
pkcs11t.h, [969](#)  
CKF\_EXCLUDE\_PIN  
pkcs11t.h, [969](#)  
CKF\_EXCLUDE\_TIME  
pkcs11t.h, [969](#)  
CKF\_EXTENSION  
pkcs11t.h, [969](#)  
CKF\_GENERATE  
pkcs11t.h, [969](#)  
CKF\_GENERATE\_KEY\_PAIR  
pkcs11t.h, [970](#)  
CKF\_HW  
pkcs11t.h, [970](#)  
CKF\_HW\_SLOT  
pkcs11t.h, [970](#)  
CKF\_LIBRARY\_CANT\_CREATE\_OS\_THREADS  
pkcs11t.h, [970](#)  
CKF\_LOGIN\_REQUIRED  
pkcs11t.h, [970](#)  
CKF\_NEXT\_OTP  
pkcs11t.h, [970](#)  
CKF\_OS\_LOCKING\_OK  
pkcs11t.h, [970](#)  
CKF\_PROTECTED\_AUTHENTICATION\_PATH  
pkcs11t.h, [970](#)  
CKF\_REMOVABLE\_DEVICE  
pkcs11t.h, [971](#)  
CKF\_RESTORE\_KEY\_NOT\_NEEDED  
pkcs11t.h, [971](#)  
CKF\_RNG  
pkcs11t.h, [971](#)  
CKF\_RW\_SESSION  
pkcs11t.h, [971](#)  
CKF\_SECONDARY\_AUTHENTICATION  
pkcs11t.h, [971](#)  
CKF\_SERIAL\_SESSION  
pkcs11t.h, [971](#)  
CKF\_SIGN  
pkcs11t.h, [971](#)  
CKF\_SIGN\_RECOVER  
pkcs11t.h, [971](#)  
CKF\_SO\_PIN\_COUNT\_LOW  
pkcs11t.h, [972](#)  
CKF\_SO\_PIN\_FINAL\_TRY  
pkcs11t.h, [972](#)  
CKF\_SO\_PIN\_LOCKED  
pkcs11t.h, [972](#)  
CKF\_SO\_PIN\_TO\_BE\_CHANGED  
pkcs11t.h, [972](#)  
CKF\_TOKEN\_INITIALIZED  
pkcs11t.h, [972](#)  
CKF\_TOKEN\_PRESENT  
pkcs11t.h, [972](#)  
CKF\_UNWRAP  
pkcs11t.h, [972](#)  
CKF\_USER\_FRIENDLY\_OTP  
pkcs11t.h, [972](#)  
CKF\_USER\_PIN\_COUNT\_LOW  
pkcs11t.h, [973](#)  
CKF\_USER\_PIN\_FINAL\_TRY  
pkcs11t.h, [973](#)  
CKF\_USER\_PIN\_INITIALIZED  
pkcs11t.h, [973](#)  
CKF\_USER\_PIN\_LOCKED  
pkcs11t.h, [973](#)  
CKF\_USER\_PIN\_TO\_BE\_CHANGED

pkcs11t.h, [973](#)  
 CKF\_VERIFY  
 pkcs11t.h, [973](#)  
 CKF\_VERIFY\_RECOVER  
 pkcs11t.h, [973](#)  
 CKF\_WRAP  
 pkcs11t.h, [973](#)  
 CKF\_WRITE\_PROTECTED  
 pkcs11t.h, [974](#)  
 CKG\_MGF1\_SHA1  
 pkcs11t.h, [974](#)  
 CKG\_MGF1\_SHA224  
 pkcs11t.h, [974](#)  
 CKG\_MGF1\_SHA256  
 pkcs11t.h, [974](#)  
 CKG\_MGF1\_SHA384  
 pkcs11t.h, [974](#)  
 CKG\_MGF1\_SHA512  
 pkcs11t.h, [974](#)  
 CKH\_CLOCK  
 pkcs11t.h, [974](#)  
 CKH\_MONOTONIC\_COUNTER  
 pkcs11t.h, [974](#)  
 CKH\_USER\_INTERFACE  
 pkcs11t.h, [975](#)  
 CKH\_VENDOR\_DEFINED  
 pkcs11t.h, [975](#)  
 CKK\_ACTI  
 pkcs11t.h, [975](#)  
 CKK\_AES  
 pkcs11t.h, [975](#)  
 CKK\_ARIA  
 pkcs11t.h, [975](#)  
 CKK\_BATON  
 pkcs11t.h, [975](#)  
 CKK\_BLOWFISH  
 pkcs11t.h, [975](#)  
 CKK\_CAMELLIA  
 pkcs11t.h, [975](#)  
 CKK\_CAST  
 pkcs11t.h, [976](#)  
 CKK\_CAST128  
 pkcs11t.h, [976](#)  
 CKK\_CAST3  
 pkcs11t.h, [976](#)  
 CKK\_CAST5  
 pkcs11t.h, [976](#)  
 CKK\_CDMF  
 pkcs11t.h, [976](#)  
 CKK\_DES  
 pkcs11t.h, [976](#)  
 CKK\_DES2  
 pkcs11t.h, [976](#)  
 CKK\_DES3  
 pkcs11t.h, [976](#)  
 CKK\_DH  
 pkcs11t.h, [977](#)  
 CKK\_DSA  
 pkcs11t.h, [977](#)  
 CKK\_EC  
 pkcs11t.h, [977](#)  
 CKK\_ECDSA  
 pkcs11t.h, [977](#)  
 CKK\_GENERIC\_SECRET  
 pkcs11t.h, [977](#)  
 CKK\_GOST28147  
 pkcs11t.h, [977](#)  
 CKK\_GOSTR3410  
 pkcs11t.h, [977](#)  
 CKK\_GOSTR3411  
 pkcs11t.h, [977](#)  
 CKK\_HOTP  
 pkcs11t.h, [978](#)  
 CKK\_IDEA  
 pkcs11t.h, [978](#)  
 CKK\_JUNIPER  
 pkcs11t.h, [978](#)  
 CKK\_KEA  
 pkcs11t.h, [978](#)  
 CKK\_MD5\_HMAC  
 pkcs11t.h, [978](#)  
 CKK\_RC2  
 pkcs11t.h, [978](#)  
 CKK\_RC4  
 pkcs11t.h, [978](#)  
 CKK\_RC5  
 pkcs11t.h, [978](#)  
 CKK\_RIPEMD128\_HMAC  
 pkcs11t.h, [979](#)  
 CKK\_RIPEMD160\_HMAC  
 pkcs11t.h, [979](#)  
 CKK\_RSA  
 pkcs11t.h, [979](#)  
 CKK\_SECURID  
 pkcs11t.h, [979](#)  
 CKK\_SEED  
 pkcs11t.h, [979](#)  
 CKK\_SHA224\_HMAC  
 pkcs11t.h, [979](#)  
 CKK\_SHA256\_HMAC  
 pkcs11t.h, [979](#)  
 CKK\_SHA384\_HMAC  
 pkcs11t.h, [979](#)  
 CKK\_SHA512\_HMAC  
 pkcs11t.h, [980](#)  
 CKK\_SHA\_1\_HMAC  
 pkcs11t.h, [980](#)  
 CKK\_SKIPJACK  
 pkcs11t.h, [980](#)  
 CKK\_TWOFISH  
 pkcs11t.h, [980](#)  
 CKK\_VENDOR\_DEFINED  
 pkcs11t.h, [980](#)  
 CKK\_X9\_42\_DH  
 pkcs11t.h, [980](#)  
 CKM\_ACTI



pkcs11t.h, [980](#)  
CKM\_ACTI\_KEY\_GEN  
pkcs11t.h, [980](#)  
CKM\_AES\_CBC  
pkcs11t.h, [981](#)  
CKM\_AES\_CBC\_ENCRYPT\_DATA  
pkcs11t.h, [981](#)  
CKM\_AES\_CBC\_PAD  
pkcs11t.h, [981](#)  
CKM\_AES\_CCM  
pkcs11t.h, [981](#)  
CKM\_AES\_CFB1  
pkcs11t.h, [981](#)  
CKM\_AES\_CFB128  
pkcs11t.h, [981](#)  
CKM\_AES\_CFB64  
pkcs11t.h, [981](#)  
CKM\_AES\_CFB8  
pkcs11t.h, [981](#)  
CKM\_AES\_CMAC  
pkcs11t.h, [982](#)  
CKM\_AES\_CMAC\_GENERAL  
pkcs11t.h, [982](#)  
CKM\_AES\_CTR  
pkcs11t.h, [982](#)  
CKM\_AES\_CTS  
pkcs11t.h, [982](#)  
CKM\_AES\_ECB  
pkcs11t.h, [982](#)  
CKM\_AES\_ECB\_ENCRYPT\_DATA  
pkcs11t.h, [982](#)  
CKM\_AES\_GCM  
pkcs11t.h, [982](#)  
CKM\_AES\_GMAC  
pkcs11t.h, [982](#)  
CKM\_AES\_KEY\_GEN  
pkcs11t.h, [983](#)  
CKM\_AES\_KEY\_WRAP  
pkcs11t.h, [983](#)  
CKM\_AES\_KEY\_WRAP\_PAD  
pkcs11t.h, [983](#)  
CKM\_AES\_MAC  
pkcs11t.h, [983](#)  
CKM\_AES\_MAC\_GENERAL  
pkcs11t.h, [983](#)  
CKM\_AES\_OFB  
pkcs11t.h, [983](#)  
CKM\_AES\_XCBC\_MAC  
pkcs11t.h, [983](#)  
CKM\_AES\_XCBC\_MAC\_96  
pkcs11t.h, [983](#)  
CKM\_ARIA\_CBC  
pkcs11t.h, [984](#)  
CKM\_ARIA\_CBC\_ENCRYPT\_DATA  
pkcs11t.h, [984](#)  
CKM\_ARIA\_CBC\_PAD  
pkcs11t.h, [984](#)  
CKM\_ARIA\_ECB  
pkcs11t.h, [984](#)  
CKM\_ARIA\_ECB\_ENCRYPT\_DATA  
pkcs11t.h, [984](#)  
CKM\_ARIA\_KEY\_GEN  
pkcs11t.h, [984](#)  
CKM\_ARIA\_MAC  
pkcs11t.h, [984](#)  
CKM\_ARIA\_MAC\_GENERAL  
pkcs11t.h, [984](#)  
CKM\_BATON\_CBC128  
pkcs11t.h, [985](#)  
CKM\_BATON\_COUNTER  
pkcs11t.h, [985](#)  
CKM\_BATON\_ECB128  
pkcs11t.h, [985](#)  
CKM\_BATON\_ECB96  
pkcs11t.h, [985](#)  
CKM\_BATON\_KEY\_GEN  
pkcs11t.h, [985](#)  
CKM\_BATON\_SHUFFLE  
pkcs11t.h, [985](#)  
CKM\_BATON\_WRAP  
pkcs11t.h, [985](#)  
CKM\_BLOWFISH\_CBC  
pkcs11t.h, [985](#)  
CKM\_BLOWFISH\_CBC\_PAD  
pkcs11t.h, [986](#)  
CKM\_BLOWFISH\_KEY\_GEN  
pkcs11t.h, [986](#)  
CKM\_CAMELLIA\_CBC  
pkcs11t.h, [986](#)  
CKM\_CAMELLIA\_CBC\_ENCRYPT\_DATA  
pkcs11t.h, [986](#)  
CKM\_CAMELLIA\_CBC\_PAD  
pkcs11t.h, [986](#)  
CKM\_CAMELLIA\_CTR  
pkcs11t.h, [986](#)  
CKM\_CAMELLIA\_ECB  
pkcs11t.h, [986](#)  
CKM\_CAMELLIA\_ECB\_ENCRYPT\_DATA  
pkcs11t.h, [986](#)  
CKM\_CAMELLIA\_KEY\_GEN  
pkcs11t.h, [987](#)  
CKM\_CAMELLIA\_MAC  
pkcs11t.h, [987](#)  
CKM\_CAMELLIA\_MAC\_GENERAL  
pkcs11t.h, [987](#)  
CKM\_CAST128\_CBC  
pkcs11t.h, [987](#)  
CKM\_CAST128\_CBC\_PAD  
pkcs11t.h, [987](#)  
CKM\_CAST128\_ECB  
pkcs11t.h, [987](#)  
CKM\_CAST128\_KEY\_GEN  
pkcs11t.h, [987](#)  
CKM\_CAST128\_MAC  
pkcs11t.h, [987](#)  
CKM\_CAST128\_MAC\_GENERAL

[pkcs11t.h, 988](#)  
 CKM\_CAST3\_CBC  
[pkcs11t.h, 988](#)  
 CKM\_CAST3\_CBC\_PAD  
[pkcs11t.h, 988](#)  
 CKM\_CAST3\_ECB  
[pkcs11t.h, 988](#)  
 CKM\_CAST3\_KEY\_GEN  
[pkcs11t.h, 988](#)  
 CKM\_CAST3\_MAC  
[pkcs11t.h, 988](#)  
 CKM\_CAST3\_MAC\_GENERAL  
[pkcs11t.h, 988](#)  
 CKM\_CAST5\_CBC  
[pkcs11t.h, 988](#)  
 CKM\_CAST5\_CBC\_PAD  
[pkcs11t.h, 989](#)  
 CKM\_CAST5\_ECB  
[pkcs11t.h, 989](#)  
 CKM\_CAST5\_KEY\_GEN  
[pkcs11t.h, 989](#)  
 CKM\_CAST5\_MAC  
[pkcs11t.h, 989](#)  
 CKM\_CAST5\_MAC\_GENERAL  
[pkcs11t.h, 989](#)  
 CKM\_CAST\_CBC  
[pkcs11t.h, 989](#)  
 CKM\_CAST\_CBC\_PAD  
[pkcs11t.h, 989](#)  
 CKM\_CAST\_ECB  
[pkcs11t.h, 989](#)  
 CKM\_CAST\_KEY\_GEN  
[pkcs11t.h, 990](#)  
 CKM\_CAST\_MAC  
[pkcs11t.h, 990](#)  
 CKM\_CAST\_MAC\_GENERAL  
[pkcs11t.h, 990](#)  
 CKM\_CDMF\_CBC  
[pkcs11t.h, 990](#)  
 CKM\_CDMF\_CBC\_PAD  
[pkcs11t.h, 990](#)  
 CKM\_CDMF\_ECB  
[pkcs11t.h, 990](#)  
 CKM\_CDMF\_KEY\_GEN  
[pkcs11t.h, 990](#)  
 CKM\_CDMF\_MAC  
[pkcs11t.h, 990](#)  
 CKM\_CDMF\_MAC\_GENERAL  
[pkcs11t.h, 991](#)  
 CKM\_CMS\_SIG  
[pkcs11t.h, 991](#)  
 CKM\_CONCATENATE\_BASE\_AND\_DATA  
[pkcs11t.h, 991](#)  
 CKM\_CONCATENATE\_BASE\_AND\_KEY  
[pkcs11t.h, 991](#)  
 CKM\_CONCATENATE\_DATA\_AND\_BASE  
[pkcs11t.h, 991](#)  
 CKM\_DES2\_KEY\_GEN  
[pkcs11t.h, 991](#)  
 CKM\_DES3\_CBC  
[pkcs11t.h, 991](#)  
 CKM\_DES3\_CBC\_ENCRYPT\_DATA  
[pkcs11t.h, 991](#)  
 CKM\_DES3\_CBC\_PAD  
[pkcs11t.h, 992](#)  
 CKM\_DES3\_CMAC  
[pkcs11t.h, 992](#)  
 CKM\_DES3\_CMAC\_GENERAL  
[pkcs11t.h, 992](#)  
 CKM\_DES3\_ECB  
[pkcs11t.h, 992](#)  
 CKM\_DES3\_ECB\_ENCRYPT\_DATA  
[pkcs11t.h, 992](#)  
 CKM\_DES3\_KEY\_GEN  
[pkcs11t.h, 992](#)  
 CKM\_DES3\_MAC  
[pkcs11t.h, 992](#)  
 CKM\_DES3\_MAC\_GENERAL  
[pkcs11t.h, 992](#)  
 CKM\_DES\_CBC  
[pkcs11t.h, 993](#)  
 CKM\_DES\_CBC\_ENCRYPT\_DATA  
[pkcs11t.h, 993](#)  
 CKM\_DES\_CBC\_PAD  
[pkcs11t.h, 993](#)  
 CKM\_DES\_CFB64  
[pkcs11t.h, 993](#)  
 CKM\_DES\_CFB8  
[pkcs11t.h, 993](#)  
 CKM\_DES\_ECB  
[pkcs11t.h, 993](#)  
 CKM\_DES\_ECB\_ENCRYPT\_DATA  
[pkcs11t.h, 993](#)  
 CKM\_DES\_KEY\_GEN  
[pkcs11t.h, 993](#)  
 CKM\_DES\_MAC  
[pkcs11t.h, 994](#)  
 CKM\_DES\_MAC\_GENERAL  
[pkcs11t.h, 994](#)  
 CKM\_DES\_OFB64  
[pkcs11t.h, 994](#)  
 CKM\_DES\_OFB8  
[pkcs11t.h, 994](#)  
 CKM\_DH\_PKCS\_DERIVE  
[pkcs11t.h, 994](#)  
 CKM\_DH\_PKCS\_KEY\_PAIR\_GEN  
[pkcs11t.h, 994](#)  
 CKM\_DH\_PKCS\_PARAMETER\_GEN  
[pkcs11t.h, 994](#)  
 CKM\_DSA  
[pkcs11t.h, 994](#)  
 CKM\_DSA\_KEY\_PAIR\_GEN  
[pkcs11t.h, 995](#)  
 CKM\_DSA\_PARAMETER\_GEN  
[pkcs11t.h, 995](#)  
 CKM\_DSA\_PROBABLISTIC\_PARAMETER\_GEN

pkcs11t.h, [995](#)  
CKM\_DSA\_SHA1  
pkcs11t.h, [995](#)  
CKM\_DSA\_SHA224  
pkcs11t.h, [995](#)  
CKM\_DSA\_SHA256  
pkcs11t.h, [995](#)  
CKM\_DSA\_SHA384  
pkcs11t.h, [995](#)  
CKM\_DSA\_SHA512  
pkcs11t.h, [995](#)  
CKM\_DSA\_SHAWA\_TAYLOR\_PARAMETER\_GEN  
pkcs11t.h, [996](#)  
CKM\_EC\_KEY\_PAIR\_GEN  
pkcs11t.h, [996](#)  
CKM\_ECDH1\_COFACTOR\_DERIVE  
pkcs11t.h, [996](#)  
CKM\_ECDH1\_DERIVE  
pkcs11t.h, [996](#)  
CKM\_ECDH\_AES\_KEY\_WRAP  
pkcs11t.h, [996](#)  
CKM\_ECDSA  
pkcs11t.h, [996](#)  
CKM\_ECDSA\_KEY\_PAIR\_GEN  
pkcs11t.h, [996](#)  
CKM\_ECDSA\_SHA1  
pkcs11t.h, [996](#)  
CKM\_ECDSA\_SHA224  
pkcs11t.h, [997](#)  
CKM\_ECDSA\_SHA256  
pkcs11t.h, [997](#)  
CKM\_ECDSA\_SHA384  
pkcs11t.h, [997](#)  
CKM\_ECDSA\_SHA512  
pkcs11t.h, [997](#)  
CKM\_ECMQV\_DERIVE  
pkcs11t.h, [997](#)  
CKM\_EXTRACT\_KEY\_FROM\_KEY  
pkcs11t.h, [997](#)  
CKM\_FASTHASH  
pkcs11t.h, [997](#)  
CKM\_FORTEZZA\_TIMESTAMP  
pkcs11t.h, [997](#)  
CKM\_GENERIC\_SECRET\_KEY\_GEN  
pkcs11t.h, [998](#)  
CKM\_GOST28147  
pkcs11t.h, [998](#)  
CKM\_GOST28147\_ECB  
pkcs11t.h, [998](#)  
CKM\_GOST28147\_KEY\_GEN  
pkcs11t.h, [998](#)  
CKM\_GOST28147\_KEY\_WRAP  
pkcs11t.h, [998](#)  
CKM\_GOST28147\_MAC  
pkcs11t.h, [998](#)  
CKM\_GOSTR3410  
pkcs11t.h, [998](#)  
CKM\_GOSTR3410\_DERIVE  
pkcs11t.h, [998](#)  
CKM\_GOSTR3410\_KEY\_PAIR\_GEN  
pkcs11t.h, [999](#)  
CKM\_GOSTR3410\_KEY\_WRAP  
pkcs11t.h, [999](#)  
CKM\_GOSTR3410\_WITH\_GOSTR3411  
pkcs11t.h, [999](#)  
CKM\_GOSTR3411  
pkcs11t.h, [999](#)  
CKM\_GOSTR3411\_HMAC  
pkcs11t.h, [999](#)  
CKM\_HOTP  
pkcs11t.h, [999](#)  
CKM\_HOTP\_KEY\_GEN  
pkcs11t.h, [999](#)  
CKM\_IDEA\_CBC  
pkcs11t.h, [999](#)  
CKM\_IDEA\_CBC\_PAD  
pkcs11t.h, [1000](#)  
CKM\_IDEA\_ECB  
pkcs11t.h, [1000](#)  
CKM\_IDEA\_KEY\_GEN  
pkcs11t.h, [1000](#)  
CKM\_IDEA\_MAC  
pkcs11t.h, [1000](#)  
CKM\_IDEA\_MAC\_GENERAL  
pkcs11t.h, [1000](#)  
CKM\_JUNIPER\_CBC128  
pkcs11t.h, [1000](#)  
CKM\_JUNIPER\_COUNTER  
pkcs11t.h, [1000](#)  
CKM\_JUNIPER\_ECB128  
pkcs11t.h, [1000](#)  
CKM\_JUNIPER\_KEY\_GEN  
pkcs11t.h, [1001](#)  
CKM\_JUNIPER\_SHUFFLE  
pkcs11t.h, [1001](#)  
CKM\_JUNIPER\_WRAP  
pkcs11t.h, [1001](#)  
CKM\_KEA\_DERIVE  
pkcs11t.h, [1001](#)  
CKM\_KEA\_KEY\_DERIVE  
pkcs11t.h, [1001](#)  
CKM\_KEA\_KEY\_PAIR\_GEN  
pkcs11t.h, [1001](#)  
CKM\_KEY\_WRAP\_LYNKS  
pkcs11t.h, [1001](#)  
CKM\_KEY\_WRAP\_SET\_OAEP  
pkcs11t.h, [1001](#)  
CKM\_KIP\_DERIVE  
pkcs11t.h, [1002](#)  
CKM\_KIP\_MAC  
pkcs11t.h, [1002](#)  
CKM\_KIP\_WRAP  
pkcs11t.h, [1002](#)  
CKM\_MD2  
pkcs11t.h, [1002](#)  
CKM\_MD2\_HMAC

[pkcs11t.h, 1002](#)  
 CKM\_MD2\_HMAC\_GENERAL  
[pkcs11t.h, 1002](#)  
 CKM\_MD2\_KEY\_DERIVATION  
[pkcs11t.h, 1002](#)  
 CKM\_MD2\_RSA\_PKCS  
[pkcs11t.h, 1002](#)  
 CKM\_MD5  
[pkcs11t.h, 1003](#)  
 CKM\_MD5\_HMAC  
[pkcs11t.h, 1003](#)  
 CKM\_MD5\_HMAC\_GENERAL  
[pkcs11t.h, 1003](#)  
 CKM\_MD5\_KEY\_DERIVATION  
[pkcs11t.h, 1003](#)  
 CKM\_MD5\_RSA\_PKCS  
[pkcs11t.h, 1003](#)  
 CKM\_PBA\_SHA1\_WITH\_SHA1\_HMAC  
[pkcs11t.h, 1003](#)  
 CKM\_PBE\_MD2\_DES\_CBC  
[pkcs11t.h, 1003](#)  
 CKM\_PBE\_MD5\_CAST128\_CBC  
[pkcs11t.h, 1003](#)  
 CKM\_PBE\_MD5\_CAST3\_CBC  
[pkcs11t.h, 1004](#)  
 CKM\_PBE\_MD5\_CAST5\_CBC  
[pkcs11t.h, 1004](#)  
 CKM\_PBE\_MD5\_CAST\_CBC  
[pkcs11t.h, 1004](#)  
 CKM\_PBE\_MD5\_DES\_CBC  
[pkcs11t.h, 1004](#)  
 CKM\_PBE\_SHA1\_CAST128\_CBC  
[pkcs11t.h, 1004](#)  
 CKM\_PBE\_SHA1\_CAST5\_CBC  
[pkcs11t.h, 1004](#)  
 CKM\_PBE\_SHA1\_DES2\_EDE\_CBC  
[pkcs11t.h, 1004](#)  
 CKM\_PBE\_SHA1\_DES3\_EDE\_CBC  
[pkcs11t.h, 1004](#)  
 CKM\_PBE\_SHA1\_RC2\_128\_CBC  
[pkcs11t.h, 1005](#)  
 CKM\_PBE\_SHA1\_RC2\_40\_CBC  
[pkcs11t.h, 1005](#)  
 CKM\_PBE\_SHA1\_RC4\_128  
[pkcs11t.h, 1005](#)  
 CKM\_PBE\_SHA1\_RC4\_40  
[pkcs11t.h, 1005](#)  
 CKM\_PKCS5\_PBKD2  
[pkcs11t.h, 1005](#)  
 CKM\_RC2\_CBC  
[pkcs11t.h, 1005](#)  
 CKM\_RC2\_CBC\_PAD  
[pkcs11t.h, 1005](#)  
 CKM\_RC2\_ECB  
[pkcs11t.h, 1005](#)  
 CKM\_RC2\_KEY\_GEN  
[pkcs11t.h, 1006](#)  
 CKM\_RC2\_MAC  
[pkcs11t.h, 1006](#)  
 CKM\_RC2\_MAC\_GENERAL  
[pkcs11t.h, 1006](#)  
 CKM\_RC4  
[pkcs11t.h, 1006](#)  
 CKM\_RC4\_KEY\_GEN  
[pkcs11t.h, 1006](#)  
 CKM\_RC5\_CBC  
[pkcs11t.h, 1006](#)  
 CKM\_RC5\_CBC\_PAD  
[pkcs11t.h, 1006](#)  
 CKM\_RC5\_ECB  
[pkcs11t.h, 1006](#)  
 CKM\_RC5\_KEY\_GEN  
[pkcs11t.h, 1007](#)  
 CKM\_RC5\_MAC  
[pkcs11t.h, 1007](#)  
 CKM\_RC5\_MAC\_GENERAL  
[pkcs11t.h, 1007](#)  
 CKM\_RIPEMD128  
[pkcs11t.h, 1007](#)  
 CKM\_RIPEMD128\_HMAC  
[pkcs11t.h, 1007](#)  
 CKM\_RIPEMD128\_HMAC\_GENERAL  
[pkcs11t.h, 1007](#)  
 CKM\_RIPEMD128\_RSA\_PKCS  
[pkcs11t.h, 1007](#)  
 CKM\_RIPEMD160  
[pkcs11t.h, 1007](#)  
 CKM\_RIPEMD160\_HMAC  
[pkcs11t.h, 1008](#)  
 CKM\_RIPEMD160\_HMAC\_GENERAL  
[pkcs11t.h, 1008](#)  
 CKM\_RIPEMD160\_RSA\_PKCS  
[pkcs11t.h, 1008](#)  
 CKM\_RSA\_9796  
[pkcs11t.h, 1008](#)  
 CKM\_RSA\_AES\_KEY\_WRAP  
[pkcs11t.h, 1008](#)  
 CKM\_RSA\_PKCS  
[pkcs11t.h, 1008](#)  
 CKM\_RSA\_PKCS\_KEY\_PAIR\_GEN  
[pkcs11t.h, 1008](#)  
 CKM\_RSA\_PKCS\_OAEP  
[pkcs11t.h, 1008](#)  
 CKM\_RSA\_PKCS\_OAEP\_TPM\_1\_1  
[pkcs11t.h, 1009](#)  
 CKM\_RSA\_PKCS\_PSS  
[pkcs11t.h, 1009](#)  
 CKM\_RSA\_PKCS\_TPM\_1\_1  
[pkcs11t.h, 1009](#)  
 CKM\_RSA\_X9\_31  
[pkcs11t.h, 1009](#)  
 CKM\_RSA\_X9\_31\_KEY\_PAIR\_GEN  
[pkcs11t.h, 1009](#)  
 CKM\_RSA\_X\_509  
[pkcs11t.h, 1009](#)  
 CKM\_SECURID

pkcs11t.h, [1009](#)  
CKM\_SECURID\_KEY\_GEN  
pkcs11t.h, [1009](#)  
CKM\_SEED\_CBC  
pkcs11t.h, [1010](#)  
CKM\_SEED\_CBC\_ENCRYPT\_DATA  
pkcs11t.h, [1010](#)  
CKM\_SEED\_CBC\_PAD  
pkcs11t.h, [1010](#)  
CKM\_SEED\_ECB  
pkcs11t.h, [1010](#)  
CKM\_SEED\_ECB\_ENCRYPT\_DATA  
pkcs11t.h, [1010](#)  
CKM\_SEED\_KEY\_GEN  
pkcs11t.h, [1010](#)  
CKM\_SEED\_MAC  
pkcs11t.h, [1010](#)  
CKM\_SEED\_MAC\_GENERAL  
pkcs11t.h, [1010](#)  
CKM\_SHA1\_KEY\_DERIVATION  
pkcs11t.h, [1011](#)  
CKM\_SHA1\_RSA\_PKCS  
pkcs11t.h, [1011](#)  
CKM\_SHA1\_RSA\_PKCS\_PSS  
pkcs11t.h, [1011](#)  
CKM\_SHA1\_RSA\_X9\_31  
pkcs11t.h, [1011](#)  
CKM\_SHA224  
pkcs11t.h, [1011](#)  
CKM\_SHA224\_HMAC  
pkcs11t.h, [1011](#)  
CKM\_SHA224\_HMAC\_GENERAL  
pkcs11t.h, [1011](#)  
CKM\_SHA224\_KEY\_DERIVATION  
pkcs11t.h, [1011](#)  
CKM\_SHA224\_RSA\_PKCS  
pkcs11t.h, [1012](#)  
CKM\_SHA224\_RSA\_PKCS\_PSS  
pkcs11t.h, [1012](#)  
CKM\_SHA256  
pkcs11t.h, [1012](#)  
CKM\_SHA256\_HMAC  
pkcs11t.h, [1012](#)  
CKM\_SHA256\_HMAC\_GENERAL  
pkcs11t.h, [1012](#)  
CKM\_SHA256\_KEY\_DERIVATION  
pkcs11t.h, [1012](#)  
CKM\_SHA256\_RSA\_PKCS  
pkcs11t.h, [1012](#)  
CKM\_SHA256\_RSA\_PKCS\_PSS  
pkcs11t.h, [1012](#)  
CKM\_SHA384  
pkcs11t.h, [1013](#)  
CKM\_SHA384\_HMAC  
pkcs11t.h, [1013](#)  
CKM\_SHA384\_HMAC\_GENERAL  
pkcs11t.h, [1013](#)  
CKM\_SHA384\_KEY\_DERIVATION  
pkcs11t.h, [1013](#)  
CKM\_SHA384\_RSA\_PKCS  
pkcs11t.h, [1013](#)  
CKM\_SHA384\_RSA\_PKCS\_PSS  
pkcs11t.h, [1013](#)  
CKM\_SHA512  
pkcs11t.h, [1013](#)  
CKM\_SHA512\_224  
pkcs11t.h, [1013](#)  
CKM\_SHA512\_224\_HMAC  
pkcs11t.h, [1014](#)  
CKM\_SHA512\_224\_HMAC\_GENERAL  
pkcs11t.h, [1014](#)  
CKM\_SHA512\_224\_KEY\_DERIVATION  
pkcs11t.h, [1014](#)  
CKM\_SHA512\_256  
pkcs11t.h, [1014](#)  
CKM\_SHA512\_256\_HMAC  
pkcs11t.h, [1014](#)  
CKM\_SHA512\_256\_HMAC\_GENERAL  
pkcs11t.h, [1014](#)  
CKM\_SHA512\_256\_KEY\_DERIVATION  
pkcs11t.h, [1014](#)  
CKM\_SHA512\_HMAC  
pkcs11t.h, [1014](#)  
CKM\_SHA512\_HMAC\_GENERAL  
pkcs11t.h, [1015](#)  
CKM\_SHA512\_KEY\_DERIVATION  
pkcs11t.h, [1015](#)  
CKM\_SHA512\_RSA\_PKCS  
pkcs11t.h, [1015](#)  
CKM\_SHA512\_RSA\_PKCS\_PSS  
pkcs11t.h, [1015](#)  
CKM\_SHA512\_T  
pkcs11t.h, [1015](#)  
CKM\_SHA512\_T\_HMAC  
pkcs11t.h, [1015](#)  
CKM\_SHA512\_T\_HMAC\_GENERAL  
pkcs11t.h, [1015](#)  
CKM\_SHA512\_T\_KEY\_DERIVATION  
pkcs11t.h, [1015](#)  
CKM\_SHA\_1  
pkcs11t.h, [1016](#)  
CKM\_SHA\_1\_HMAC  
pkcs11t.h, [1016](#)  
CKM\_SHA\_1\_HMAC\_GENERAL  
pkcs11t.h, [1016](#)  
CKM\_SKIPJACK\_CBC64  
pkcs11t.h, [1016](#)  
CKM\_SKIPJACK\_CFB16  
pkcs11t.h, [1016](#)  
CKM\_SKIPJACK\_CFB32  
pkcs11t.h, [1016](#)  
CKM\_SKIPJACK\_CFB64  
pkcs11t.h, [1016](#)  
CKM\_SKIPJACK\_CFB8  
pkcs11t.h, [1016](#)  
CKM\_SKIPJACK\_ECB64

pkcs11t.h, [1017](#)  
 CKM\_SKIPJACK\_KEY\_GEN  
 pkcs11t.h, [1017](#)  
 CKM\_SKIPJACK\_OFB64  
 pkcs11t.h, [1017](#)  
 CKM\_SKIPJACK\_PRIVATE\_WRAP  
 pkcs11t.h, [1017](#)  
 CKM\_SKIPJACK\_RELAYX  
 pkcs11t.h, [1017](#)  
 CKM\_SKIPJACK\_WRAP  
 pkcs11t.h, [1017](#)  
 CKM\_SSL3\_KEY\_AND\_MAC\_DERIVE  
 pkcs11t.h, [1017](#)  
 CKM\_SSL3\_MASTER\_KEY\_DERIVE  
 pkcs11t.h, [1017](#)  
 CKM\_SSL3\_MASTER\_KEY\_DERIVE\_DH  
 pkcs11t.h, [1018](#)  
 CKM\_SSL3\_MD5\_MAC  
 pkcs11t.h, [1018](#)  
 CKM\_SSL3\_PRE\_MASTER\_KEY\_GEN  
 pkcs11t.h, [1018](#)  
 CKM\_SSL3\_SHA1\_MAC  
 pkcs11t.h, [1018](#)  
 CKM\_TLS10\_MAC\_CLIENT  
 pkcs11t.h, [1018](#)  
 CKM\_TLS10\_MAC\_SERVER  
 pkcs11t.h, [1018](#)  
 CKM\_TLS12\_KDF  
 pkcs11t.h, [1018](#)  
 CKM\_TLS12\_KEY\_AND\_MAC\_DERIVE  
 pkcs11t.h, [1018](#)  
 CKM\_TLS12\_KEY\_SAFE\_DERIVE  
 pkcs11t.h, [1019](#)  
 CKM\_TLS12\_MAC  
 pkcs11t.h, [1019](#)  
 CKM\_TLS12\_MASTER\_KEY\_DERIVE  
 pkcs11t.h, [1019](#)  
 CKM\_TLS12\_MASTER\_KEY\_DERIVE\_DH  
 pkcs11t.h, [1019](#)  
 CKM\_TLS\_KDF  
 pkcs11t.h, [1019](#)  
 CKM\_TLS\_KEY\_AND\_MAC\_DERIVE  
 pkcs11t.h, [1019](#)  
 CKM\_TLS\_MAC  
 pkcs11t.h, [1019](#)  
 CKM\_TLS\_MASTER\_KEY\_DERIVE  
 pkcs11t.h, [1019](#)  
 CKM\_TLS\_MASTER\_KEY\_DERIVE\_DH  
 pkcs11t.h, [1020](#)  
 CKM\_TLS\_PRE\_MASTER\_KEY\_GEN  
 pkcs11t.h, [1020](#)  
 CKM\_TLS\_PRFB  
 pkcs11t.h, [1020](#)  
 CKM\_TWOFISH\_CBC  
 pkcs11t.h, [1020](#)  
 CKM\_TWOFISH\_CBC\_PAD  
 pkcs11t.h, [1020](#)  
 CKM\_TWOFISH\_KEY\_GEN  
 pkcs11t.h, [1020](#)  
 CKM\_VENDOR\_DEFINED  
 pkcs11t.h, [1020](#)  
 CKM\_WTLS\_CLIENT\_KEY\_AND\_MAC\_DERIVE  
 pkcs11t.h, [1020](#)  
 CKM\_WTLS\_MASTER\_KEY\_DERIVE  
 pkcs11t.h, [1021](#)  
 CKM\_WTLS\_MASTER\_KEY\_DERIVE\_DH\_ECC  
 pkcs11t.h, [1021](#)  
 CKM\_WTLS\_PRE\_MASTER\_KEY\_GEN  
 pkcs11t.h, [1021](#)  
 CKM\_WTLS\_PRFB  
 pkcs11t.h, [1021](#)  
 CKM\_WTLS\_SERVER\_KEY\_AND\_MAC\_DERIVE  
 pkcs11t.h, [1021](#)  
 CKM\_X9\_42\_DH\_DERIVE  
 pkcs11t.h, [1021](#)  
 CKM\_X9\_42\_DH\_HYBRID\_DERIVE  
 pkcs11t.h, [1021](#)  
 CKM\_X9\_42\_DH\_KEY\_PAIR\_GEN  
 pkcs11t.h, [1021](#)  
 CKM\_X9\_42\_DH\_PARAMETER\_GEN  
 pkcs11t.h, [1022](#)  
 CKM\_X9\_42\_MQV\_DERIVE  
 pkcs11t.h, [1022](#)  
 CKM\_XOR\_BASE\_AND\_DATA  
 pkcs11t.h, [1022](#)  
 CKN\_OTP\_CHANGED  
 pkcs11t.h, [1022](#)  
 CKN\_SURRENDER  
 pkcs11t.h, [1022](#)  
 CKO\_CERTIFICATE  
 pkcs11t.h, [1022](#)  
 CKO\_DATA  
 pkcs11t.h, [1022](#)  
 CKO\_DOMAIN\_PARAMETERS  
 pkcs11t.h, [1022](#)  
 CKO\_HW\_FEATURE  
 pkcs11t.h, [1023](#)  
 CKO\_MECHANISM  
 pkcs11t.h, [1023](#)  
 CKO\_OTP\_KEY  
 pkcs11t.h, [1023](#)  
 CKO\_PRIVATE\_KEY  
 pkcs11t.h, [1023](#)  
 CKO\_PUBLIC\_KEY  
 pkcs11t.h, [1023](#)  
 CKO\_SECRET\_KEY  
 pkcs11t.h, [1023](#)  
 CKO\_VENDOR\_DEFINED  
 pkcs11t.h, [1023](#)  
 CKP\_PKCS5\_PBKD2\_HMAC\_GOSTR3411  
 pkcs11t.h, [1023](#)  
 CKP\_PKCS5\_PBKD2\_HMAC\_SHA1  
 pkcs11t.h, [1024](#)  
 CKP\_PKCS5\_PBKD2\_HMAC\_SHA224  
 pkcs11t.h, [1024](#)  
 CKP\_PKCS5\_PBKD2\_HMAC\_SHA256

pkcs11t.h, [1024](#)  
CKP\_PKCS5\_PBKD2\_HMAC\_SHA384  
pkcs11t.h, [1024](#)  
CKP\_PKCS5\_PBKD2\_HMAC\_SHA512  
pkcs11t.h, [1024](#)  
CKP\_PKCS5\_PBKD2\_HMAC\_SHA512\_224  
pkcs11t.h, [1024](#)  
CKP\_PKCS5\_PBKD2\_HMAC\_SHA512\_256  
pkcs11t.h, [1024](#)  
CKR\_ACTION\_PROHIBITED  
pkcs11t.h, [1024](#)  
CKR\_ARGUMENTS\_BAD  
pkcs11t.h, [1025](#)  
CKR\_ATTRIBUTE\_READ\_ONLY  
pkcs11t.h, [1025](#)  
CKR\_ATTRIBUTE\_SENSITIVE  
pkcs11t.h, [1025](#)  
CKR\_ATTRIBUTE\_TYPE\_INVALID  
pkcs11t.h, [1025](#)  
CKR\_ATTRIBUTE\_VALUE\_INVALID  
pkcs11t.h, [1025](#)  
CKR\_BUFFER\_TOO\_SMALL  
pkcs11t.h, [1025](#)  
CKR\_CANCEL  
pkcs11t.h, [1025](#)  
CKR\_CANT\_LOCK  
pkcs11t.h, [1025](#)  
CKR\_CRYPTOKI\_ALREADY\_INITIALIZED  
pkcs11t.h, [1026](#)  
CKR\_CRYPTOKI\_NOT\_INITIALIZED  
pkcs11t.h, [1026](#)  
CKR\_CURVE\_NOT\_SUPPORTED  
pkcs11t.h, [1026](#)  
CKR\_DATA\_INVALID  
pkcs11t.h, [1026](#)  
CKR\_DATA\_LEN\_RANGE  
pkcs11t.h, [1026](#)  
CKR\_DEVICE\_ERROR  
pkcs11t.h, [1026](#)  
CKR\_DEVICE\_MEMORY  
pkcs11t.h, [1026](#)  
CKR\_DEVICE\_REMOVED  
pkcs11t.h, [1026](#)  
CKR\_DOMAIN\_PARAMS\_INVALID  
pkcs11t.h, [1027](#)  
CKR\_ENCRYPTED\_DATA\_INVALID  
pkcs11t.h, [1027](#)  
CKR\_ENCRYPTED\_DATA\_LEN\_RANGE  
pkcs11t.h, [1027](#)  
CKR\_EXCEEDED\_MAX\_ITERATIONS  
pkcs11t.h, [1027](#)  
CKR\_FIPS\_SELF\_TEST\_FAILED  
pkcs11t.h, [1027](#)  
CKR\_FUNCTION\_CANCELED  
pkcs11t.h, [1027](#)  
CKR\_FUNCTION\_FAILED  
pkcs11t.h, [1027](#)  
CKR\_FUNCTION\_NOT\_PARALLEL  
pkcs11t.h, [1027](#)  
CKR\_FUNCTION\_NOT\_SUPPORTED  
pkcs11t.h, [1028](#)  
CKR\_FUNCTION\_REJECTED  
pkcs11t.h, [1028](#)  
CKR\_GENERAL\_ERROR  
pkcs11t.h, [1028](#)  
CKR\_HOST\_MEMORY  
pkcs11t.h, [1028](#)  
CKR\_INFORMATION\_SENSITIVE  
pkcs11t.h, [1028](#)  
CKR\_KEY\_CHANGED  
pkcs11t.h, [1028](#)  
CKR\_KEY\_FUNCTION\_NOT\_PERMITTED  
pkcs11t.h, [1028](#)  
CKR\_KEY\_HANDLE\_INVALID  
pkcs11t.h, [1028](#)  
CKR\_KEY\_INDIGESTIBLE  
pkcs11t.h, [1029](#)  
CKR\_KEY\_NEEDED  
pkcs11t.h, [1029](#)  
CKR\_KEY\_NOT\_NEEDED  
pkcs11t.h, [1029](#)  
CKR\_KEY\_NOT\_WRAPPABLE  
pkcs11t.h, [1029](#)  
CKR\_KEY\_SIZE\_RANGE  
pkcs11t.h, [1029](#)  
CKR\_KEY\_TYPE\_INCONSISTENT  
pkcs11t.h, [1029](#)  
CKR\_KEY\_UNEXTRACTABLE  
pkcs11t.h, [1029](#)  
CKR\_LIBRARY\_LOAD\_FAILED  
pkcs11t.h, [1029](#)  
CKR\_MECHANISM\_INVALID  
pkcs11t.h, [1030](#)  
CKR\_MECHANISM\_PARAM\_INVALID  
pkcs11t.h, [1030](#)  
CKR\_MUTEX\_BAD  
pkcs11t.h, [1030](#)  
CKR\_MUTEX\_NOT\_LOCKED  
pkcs11t.h, [1030](#)  
CKR\_NEED\_TO\_CREATE\_THREADS  
pkcs11t.h, [1030](#)  
CKR\_NEW\_PIN\_MODE  
pkcs11t.h, [1030](#)  
CKR\_NEXT\_OTP  
pkcs11t.h, [1030](#)  
CKR\_NO\_EVENT  
pkcs11t.h, [1030](#)  
CKR\_OBJECT\_HANDLE\_INVALID  
pkcs11t.h, [1031](#)  
CKR\_OK  
pkcs11t.h, [1031](#)  
CKR\_OPERATION\_ACTIVE  
pkcs11t.h, [1031](#)  
CKR\_OPERATION\_NOT\_INITIALIZED  
pkcs11t.h, [1031](#)  
CKR\_PIN\_EXPIRED

[pkcs11t.h](#), [1031](#)  
 CKR\_PIN\_INCORRECT  
     [pkcs11t.h](#), [1031](#)  
 CKR\_PIN\_INVALID  
     [pkcs11t.h](#), [1031](#)  
 CKR\_PIN\_LEN\_RANGE  
     [pkcs11t.h](#), [1031](#)  
 CKR\_PIN\_LOCKED  
     [pkcs11t.h](#), [1032](#)  
 CKR\_PIN\_TOO\_WEAK  
     [pkcs11t.h](#), [1032](#)  
 CKR\_PUBLIC\_KEY\_INVALID  
     [pkcs11t.h](#), [1032](#)  
 CKR\_RANDOM\_NO\_RNG  
     [pkcs11t.h](#), [1032](#)  
 CKR\_RANDOM\_SEED\_NOT\_SUPPORTED  
     [pkcs11t.h](#), [1032](#)  
 CKR\_SAVED\_STATE\_INVALID  
     [pkcs11t.h](#), [1032](#)  
 CKR\_SESSION\_CLOSED  
     [pkcs11t.h](#), [1032](#)  
 CKR\_SESSION\_COUNT  
     [pkcs11t.h](#), [1032](#)  
 CKR\_SESSION\_EXISTS  
     [pkcs11t.h](#), [1033](#)  
 CKR\_SESSION\_HANDLE\_INVALID  
     [pkcs11t.h](#), [1033](#)  
 CKR\_SESSION\_PARALLEL\_NOT\_SUPPORTED  
     [pkcs11t.h](#), [1033](#)  
 CKR\_SESSION\_READ\_ONLY  
     [pkcs11t.h](#), [1033](#)  
 CKR\_SESSION\_READ\_ONLY\_EXISTS  
     [pkcs11t.h](#), [1033](#)  
 CKR\_SESSION\_READ\_WRITE\_SO\_EXISTS  
     [pkcs11t.h](#), [1033](#)  
 CKR\_SIGNATURE\_INVALID  
     [pkcs11t.h](#), [1033](#)  
 CKR\_SIGNATURE\_LEN\_RANGE  
     [pkcs11t.h](#), [1033](#)  
 CKR\_SLOT\_ID\_INVALID  
     [pkcs11t.h](#), [1034](#)  
 CKR\_STATE\_UNSAVEABLE  
     [pkcs11t.h](#), [1034](#)  
 CKR\_TEMPLATE\_INCOMPLETE  
     [pkcs11t.h](#), [1034](#)  
 CKR\_TEMPLATE\_INCONSISTENT  
     [pkcs11t.h](#), [1034](#)  
 CKR\_TOKEN\_NOT\_PRESENT  
     [pkcs11t.h](#), [1034](#)  
 CKR\_TOKEN\_NOT\_RECOGNIZED  
     [pkcs11t.h](#), [1034](#)  
 CKR\_TOKEN\_WRITE\_PROTECTED  
     [pkcs11t.h](#), [1034](#)  
 CKR\_UNWRAPPING\_KEY\_HANDLE\_INVALID  
     [pkcs11t.h](#), [1034](#)  
 CKR\_UNWRAPPING\_KEY\_SIZE\_RANGE  
     [pkcs11t.h](#), [1035](#)  
 CKR\_UNWRAPPING\_KEY\_TYPE\_INCONSISTENT  
     [pkcs11t.h](#), [1035](#)  
 CKR\_USER\_ALREADY\_LOGGED\_IN  
     [pkcs11t.h](#), [1035](#)  
 CKR\_USER\_ANOTHER\_ALREADY\_LOGGED\_IN  
     [pkcs11t.h](#), [1035](#)  
 CKR\_USER\_NOT\_LOGGED\_IN  
     [pkcs11t.h](#), [1035](#)  
 CKR\_USER\_PIN\_NOT\_INITIALIZED  
     [pkcs11t.h](#), [1035](#)  
 CKR\_USER\_TOO\_MANY\_TYPES  
     [pkcs11t.h](#), [1035](#)  
 CKR\_USER\_TYPE\_INVALID  
     [pkcs11t.h](#), [1035](#)  
 CKR\_VENDOR\_DEFINED  
     [pkcs11t.h](#), [1036](#)  
 CKR\_WRAPPED\_KEY\_INVALID  
     [pkcs11t.h](#), [1036](#)  
 CKR\_WRAPPED\_KEY\_LEN\_RANGE  
     [pkcs11t.h](#), [1036](#)  
 CKR\_WRAPPING\_KEY\_HANDLE\_INVALID  
     [pkcs11t.h](#), [1036](#)  
 CKR\_WRAPPING\_KEY\_SIZE\_RANGE  
     [pkcs11t.h](#), [1036](#)  
 CKR\_WRAPPING\_KEY\_TYPE\_INCONSISTENT  
     [pkcs11t.h](#), [1036](#)  
 CKS\_RO\_PUBLIC\_SESSION  
     [pkcs11t.h](#), [1036](#)  
 CKS\_RO\_USER\_FUNCTIONS  
     [pkcs11t.h](#), [1036](#)  
 CKS\_RW\_PUBLIC\_SESSION  
     [pkcs11t.h](#), [1037](#)  
 CKS\_RW\_SO\_FUNCTIONS  
     [pkcs11t.h](#), [1037](#)  
 CKS\_RW\_USER\_FUNCTIONS  
     [pkcs11t.h](#), [1037](#)  
 CKU\_CONTEXT\_SPECIFIC  
     [pkcs11t.h](#), [1037](#)  
 CKU\_SO  
     [pkcs11t.h](#), [1037](#)  
 CKU\_USER  
     [pkcs11t.h](#), [1037](#)  
 CKZ\_DATA\_SPECIFIED  
     [pkcs11t.h](#), [1037](#)  
 CKZ\_SALT\_SPECIFIED  
     [pkcs11t.h](#), [1037](#)  
 CL\_hash  
     [sha1\\_routines.c](#), [1069](#)  
     [sha1\\_routines.h](#), [1073](#)  
 CL\_HashContext, [533](#)  
     buf, [534](#)  
     byteCount, [534](#)  
     byteCountHi, [534](#)  
     h, [534](#)  
 CL\_hashFinal  
     [sha1\\_routines.c](#), [1070](#)  
     [sha1\\_routines.h](#), [1073](#)  
 CL\_hashInit  
     [sha1\\_routines.c](#), [1070](#)



- sha1\_routines.h, 1074
- CL\_hashUpdate
  - sha1\_routines.c, 1070
  - sha1\_routines.h, 1074
- class\_id
  - \_pkcs11\_object, 401
- class\_type
  - \_pkcs11\_object, 401
- client\_chal
  - atca\_check\_mac\_in\_out, 414
- client\_resp
  - atca\_check\_mac\_in\_out, 414
- clock\_divider
  - atca\_command, 416
- CMD\_STATUS\_BYTE\_COMM
  - calib\_command.h, 736
- CMD\_STATUS\_BYTE\_ECC
  - calib\_command.h, 736
- CMD\_STATUS\_BYTE\_EXEC
  - calib\_command.h, 736
- CMD\_STATUS\_BYTE\_PARSE
  - calib\_command.h, 736
- CMD\_STATUS\_SUCCESS
  - calib\_command.h, 737
- CMD\_STATUS\_WAKEUP
  - calib\_command.h, 737
- comp\_cert\_dev\_loc
  - atcacert\_def\_s, 451
- conf
  - hal\_esp32\_i2c.c, 846
- config
  - \_pkcs11\_object, 401
- config\_path
  - \_pkcs11\_lib\_ctx, 399
- Configuration (cfg\_), 121
- CONTRACT
  - license.txt, 884
- count
  - \_pkcs11\_object, 401
  - atcacert\_cert\_loc\_s, 448
  - atcacert\_device\_loc\_s, 453
- Counter
  - \_atsha204a\_config, 395
- counter
  - atca\_gen\_dig\_in\_out, 422
- Counter0
  - \_atecc508a\_config, 388
  - \_atecc608\_config, 392
- Counter1
  - \_atecc508a\_config, 388
  - \_atecc608\_config, 392
- COUNTER\_COUNT
  - calib\_command.h, 737
- COUNTER\_KEYID\_IDX
  - calib\_command.h, 737
- COUNTER\_MAX\_VALUE
  - calib\_command.h, 737
- COUNTER\_MODE\_IDX
  - calib\_command.h, 737
- COUNTER\_MODE\_INCREMENT
  - calib\_command.h, 738
- COUNTER\_MODE\_MASK
  - calib\_command.h, 738
- COUNTER\_MODE\_READ
  - calib\_command.h, 738
- COUNTER\_RSP\_SIZE
  - calib\_command.h, 738
- COUNTER\_SIZE
  - calib\_command.h, 738
- counter\_size
  - atca\_aes\_ctr\_ctx, 409
- CountMatch
  - \_atecc608\_config, 392
- create\_mutex
  - \_pkcs11\_lib\_ctx, 399
- CreateMutex
  - CK\_C\_INITIALIZE\_ARGS, 472
- crypto\_data
  - Host side crypto methods (atcah\_), 326
- CRYPTOAUTH\_ROOT\_CA\_002\_PUBLIC\_KEY\_OFFSET
  - TNG API (tng\_), 379
- cryptoauthlib.h, 822
  - ATCA\_AES128\_BLOCK\_SIZE, 823
  - ATCA\_AES128\_KEY\_SIZE, 823
  - ATCA\_ATECC608\_SUPPORT, 824
  - ATCA\_CA\_SUPPORT, 824
  - ATCA\_DLL, 824
  - ATCA\_ECC\_SUPPORT, 824
  - ATCA\_ECCP256\_KEY\_SIZE, 824
  - ATCA\_ECCP256\_PUBKEY\_SIZE, 824
  - ATCA\_ECCP256\_SIG\_SIZE, 824
  - ATCA\_SHA256\_BLOCK\_SIZE, 824
  - ATCA\_SHA256\_DIGEST\_SIZE, 825
  - ATCA\_SHA\_SUPPORT, 825
  - ATCA\_STRINGIFY, 825
  - ATCA\_TA\_SUPPORT, 825
  - ATCA\_TOSTRING, 825
  - ATCA\_TRACE, 825
  - ATCA\_ZONE\_CONFIG, 825
  - ATCA\_ZONE\_DATA, 826
  - ATCA\_ZONE\_OTP, 826
  - SHA\_MODE\_TARGET\_MSGDIGBUF, 826
  - SHA\_MODE\_TARGET\_OUT\_ONLY, 826
  - SHA\_MODE\_TARGET\_TEMPKEY, 826
- cryptoki.h, 826
  - CK\_CALLBACK\_FUNCTION, 827
  - CK\_DECLARE\_FUNCTION, 827
  - CK\_DECLARE\_FUNCTION\_POINTER, 827
  - CK\_PTR, 827
  - NULL\_PTR, 827
  - PKCS11\_API, 828
  - PKCS11\_HELPER\_DLL\_EXPORT, 828
  - PKCS11\_HELPER\_DLL\_IMPORT, 828
  - PKCS11\_HELPER\_DLL\_LOCAL, 828
  - PKCS11\_LOCAL, 828
- CRYPTOKI\_VERSION\_AMENDMENT

- pkcs11t.h, [1038](#)
- CRYPTOKI\_VERSION\_MAJOR
  - pkcs11t.h, [1038](#)
- CRYPTOKI\_VERSION\_MINOR
  - pkcs11t.h, [1038](#)
- cryptokiVersion
  - CK\_INFO, [488](#)
- cur
  - atca\_jwt\_t, [430](#)
- curve\_type
  - Host side crypto methods (atcah\_), [326](#)
- DAMAGE
  - license.txt, [885](#)
- DAMAGES
  - license.txt, [882](#)
- data
  - \_pkcs11\_object, [401](#)
  - atca\_io\_decrypt\_in\_out, [429](#)
  - ATCAPacket, [465](#)
- data\_size
  - atca\_aes\_gcm\_ctx, [412](#)
  - atca\_io\_decrypt\_in\_out, [429](#)
- DATEFMT\_ISO8601\_SEP
  - Certificate manipulation methods (atcacert\_), [165](#)
- DATEFMT\_ISO8601\_SEP\_SIZE
  - Certificate manipulation methods (atcacert\_), [161](#)
- DATEFMT\_MAX\_SIZE
  - Certificate manipulation methods (atcacert\_), [161](#)
- DATEFMT\_POSIX\_UINT32\_BE
  - Certificate manipulation methods (atcacert\_), [165](#)
- DATEFMT\_POSIX\_UINT32\_BE\_SIZE
  - Certificate manipulation methods (atcacert\_), [161](#)
- DATEFMT\_POSIX\_UINT32\_LE
  - Certificate manipulation methods (atcacert\_), [166](#)
- DATEFMT\_POSIX\_UINT32\_LE\_SIZE
  - Certificate manipulation methods (atcacert\_), [161](#)
- DATEFMT\_RFC5280\_GEN
  - Certificate manipulation methods (atcacert\_), [166](#)
- DATEFMT\_RFC5280\_GEN\_SIZE
  - Certificate manipulation methods (atcacert\_), [161](#)
- DATEFMT\_RFC5280\_UTC
  - Certificate manipulation methods (atcacert\_), [165](#)
- DATEFMT\_RFC5280\_UTC\_SIZE
  - Certificate manipulation methods (atcacert\_), [161](#)
- day
  - CK\_DATE, [477](#)
- DEBUG\_PIN\_1
  - Hardware abstraction layer (hal\_), [262](#)
- DEBUG\_PIN\_2
  - Hardware abstraction layer (hal\_), [263](#)
- deleteATCACommand
  - ATCACommand (atca\_), [122](#)
- deleteATCADevice
  - ATCADevice (atca\_), [144](#)
- deleteATCAIface
  - ATCAIface (atca\_), [151](#)
- DERIVE\_KEY\_COUNT\_LARGE
  - calib\_command.h, [738](#)
- DERIVE\_KEY\_COUNT\_SMALL
  - calib\_command.h, [739](#)
- DERIVE\_KEY\_MAC\_IDX
  - calib\_command.h, [739](#)
- DERIVE\_KEY\_MAC\_SIZE
  - calib\_command.h, [739](#)
- DERIVE\_KEY\_MODE
  - calib\_command.h, [739](#)
- DERIVE\_KEY\_RANDOM\_FLAG
  - calib\_command.h, [739](#)
- DERIVE\_KEY\_RANDOM\_IDX
  - calib\_command.h, [739](#)
- DERIVE\_KEY\_RSP\_SIZE
  - calib\_command.h, [740](#)
- DERIVE\_KEY\_TARGETKEY\_IDX
  - calib\_command.h, [740](#)
- destroy\_mutex
  - \_pkcs11\_lib\_ctx, [399](#)
- DestroyMutex
  - CK\_C\_INITIALIZE\_ARGS, [473](#)
- dev\_identity
  - ATCAIfaceCfg, [461](#)
- dev\_interface
  - ATCAIfaceCfg, [461](#)
- device
  - atca\_aes\_cbc\_ctx, [407](#)
  - atca\_aes\_ctr\_ctx, [410](#)
- device\_ctx
  - \_pkcs11\_slot\_ctx, [406](#)
- device\_loc
  - atcacert\_cert\_element\_s, [447](#)
- device\_sn
  - atcacert\_build\_state\_s, [446](#)
- devtype
  - ATCAIfaceCfg, [461](#)
- DEVZONE\_CONFIG
  - Certificate manipulation methods (atcacert\_), [166](#)
- DEVZONE\_DATA
  - Certificate manipulation methods (atcacert\_), [166](#)
- DEVZONE\_NONE
  - Certificate manipulation methods (atcacert\_), [166](#)
- DEVZONE\_OTP
  - Certificate manipulation methods (atcacert\_), [166](#)
- digest
  - atca\_secureboot\_enc\_in\_out, [432](#)
  - atca\_secureboot\_mac\_in\_out, [433](#)
  - atca\_sign\_internal\_in\_out, [436](#)
- digest\_enc
  - atca\_secureboot\_enc\_in\_out, [432](#)
- DigestMechanism
  - CK\_WTLS\_KEY\_MAT\_PARAMS, [525](#)
  - CK\_WTLS\_MASTER\_KEY\_DERIVE\_PARAMS, [526](#)
  - CK\_WTLS\_PRF\_PARAMS, [527](#)
- DIRECT
  - license.txt, [885](#)
- dt
  - atca\_command, [416](#)

- ECDH\_COUNT
  - calib\_command.h, [740](#)
- ECDH\_KEY\_SIZE
  - calib\_command.h, [740](#)
- ECDH\_MODE\_COPY\_COMPATIBLE
  - calib\_command.h, [740](#)
- ECDH\_MODE\_COPY\_EEPROM\_SLOT
  - calib\_command.h, [740](#)
- ECDH\_MODE\_COPY\_MASK
  - calib\_command.h, [740](#)
- ECDH\_MODE\_COPY\_OUTPUT\_BUFFER
  - calib\_command.h, [741](#)
- ECDH\_MODE\_COPY\_TEMP\_KEY
  - calib\_command.h, [741](#)
- ECDH\_MODE\_OUTPUT\_CLEAR
  - calib\_command.h, [741](#)
- ECDH\_MODE\_OUTPUT\_ENC
  - calib\_command.h, [741](#)
- ECDH\_MODE\_OUTPUT\_MASK
  - calib\_command.h, [741](#)
- ECDH\_MODE\_SOURCE\_EEPROM\_SLOT
  - calib\_command.h, [741](#)
- ECDH\_MODE\_SOURCE\_MASK
  - calib\_command.h, [741](#)
- ECDH\_MODE\_SOURCE\_TEMPKEY
  - calib\_command.h, [741](#)
- ECDH\_PREFIX\_MODE
  - calib\_command.h, [742](#)
- ECDH\_RSP\_SIZE
  - calib\_command.h, [742](#)
- enc\_cb
  - atca\_aes\_gcm\_ctx, [412](#)
- encrypted\_data
  - atca\_write\_mac\_in\_out, [444](#)
- ENCRYPTION\_KEY\_SIZE
  - Host side crypto methods (atcah\_), [315](#)
- error
  - \_pkcs11\_session\_ctx, [404](#)
- ets\_delay\_us
  - hal\_esp32\_timer.c, [847](#)
- event
  - pkcs11t.h, [1061](#)
- example\_cert\_chain.c, [828](#)
  - g\_cert\_def\_0\_root, [829](#)
  - g\_cert\_def\_1\_signer, [829](#)
  - g\_cert\_def\_2\_device, [829](#)
  - g\_cert\_elements\_1\_signer, [829](#)
  - g\_cert\_template\_1\_signer, [829](#)
  - g\_cert\_template\_2\_device, [829](#)
- example\_cert\_chain.h, [830](#)
  - g\_cert\_def\_1\_signer, [830](#)
  - g\_cert\_def\_2\_device, [830](#)
- example\_pkcs11\_config.c, [831](#)
  - atecc608\_config, [832](#)
  - pkcs11\_config\_cert, [832](#)
  - pkcs11\_config\_key, [832](#)
  - pkcs11\_config\_load\_objects, [832](#)
  - pkcs11configLABEL\_DEVICE\_CERTIFICATE\_FOR\_TLS, [831](#)
  - pkcs11configLABEL\_DEVICE\_PRIVATE\_KEY\_FOR\_TLS, [831](#)
  - pkcs11configLABEL\_DEVICE\_PUBLIC\_KEY\_FOR\_TLS, [831](#)
  - pkcs11configLABEL\_JITP\_CERTIFICATE, [832](#)
- execTime
  - ATCAPacket, [465](#)
- execution\_time\_msec
  - atca\_command, [416](#)
- EXEMPLARY
  - license.txt, [886](#)
- expire\_date\_format
  - atcacert\_def\_s, [451](#)
- expire\_years
  - atcacert\_def\_s, [451](#)
- EXPRESS
  - license.txt, [886](#)
- FALSE
  - Certificate manipulation methods (atcacert\_), [162](#)
  - pkcs11t.h, [1038](#)
- FEES
  - license.txt, [886](#)
- firmwareVersion
  - CK\_SLOT\_INFO, [511](#)
  - CK\_TOKEN\_INFO, [521](#)
- flags
  - \_pkcs11\_object, [402](#)
  - \_pkcs11\_slot\_ctx, [406](#)
  - CK\_C\_INITIALIZE\_ARGS, [473](#)
  - CK\_INFO, [488](#)
  - CK\_MECHANISM\_INFO, [493](#)
  - CK\_SESSION\_INFO, [506](#)
  - CK\_SLOT\_INFO, [511](#)
  - CK\_TOKEN\_INFO, [521](#)
- for\_invalidate
  - atca\_sign\_internal\_in\_out, [436](#)
- forms
  - license.txt, [886](#)
- Foundation
  - license.txt, [887](#)
- func
  - \_pkcs11\_attr\_model, [398](#)
- g\_cert\_def\_0\_root
  - example\_cert\_chain.c, [829](#)
- g\_cert\_def\_1\_signer
  - example\_cert\_chain.c, [829](#)
  - example\_cert\_chain.h, [830](#)
- g\_cert\_def\_2\_device
  - example\_cert\_chain.c, [829](#)
  - example\_cert\_chain.h, [830](#)
- g\_cert\_elements\_1\_signer
  - example\_cert\_chain.c, [829](#)
- g\_cert\_template\_1\_signer
  - example\_cert\_chain.c, [829](#)
- g\_cert\_template\_2\_device

- example\_cert\_chain.c, [829](#)
- g\_cryptoauth\_root\_ca\_002\_cert
  - TNG API (tng\_), [384](#)
  - tng\_root\_cert.c, [1093](#)
- g\_cryptoauth\_root\_ca\_002\_cert\_size
  - TNG API (tng\_), [384](#)
  - tng\_root\_cert.c, [1094](#)
- g\_tflxtls\_cert\_def\_4\_device
  - TNG API (tng\_), [384](#)
- g\_tflxtls\_cert\_elements\_4\_device
  - tflxtls\_cert\_def\_4\_device.c, [1086](#)
- g\_tflxtls\_cert\_template\_4\_device
  - tflxtls\_cert\_def\_4\_device.c, [1086](#)
- g\_tnglora\_cert\_def\_1\_signer
  - TNG API (tng\_), [384](#)
- g\_tnglora\_cert\_def\_2\_device
  - TNG API (tng\_), [384](#)
- g\_tnglora\_cert\_def\_4\_device
  - TNG API (tng\_), [385](#)
  - tnglora\_cert\_def\_4\_device.c, [1097](#)
- g\_tnglora\_cert\_elements\_4\_device
  - tnglora\_cert\_def\_4\_device.c, [1097](#)
- g\_tnglora\_cert\_template\_4\_device
  - tnglora\_cert\_def\_4\_device.c, [1098](#)
- g\_tngtls\_cert\_def\_1\_signer
  - TNG API (tng\_), [385](#)
  - tngtls\_cert\_def\_1\_signer.c, [1099](#)
- g\_tngtls\_cert\_def\_2\_device
  - TNG API (tng\_), [385](#)
  - tngtls\_cert\_def\_2\_device.c, [1100](#)
- g\_tngtls\_cert\_def\_3\_device
  - TNG API (tng\_), [385](#)
  - tngtls\_cert\_def\_3\_device.c, [1102](#)
- g\_tngtls\_cert\_elements\_1\_signer
  - tnglora\_cert\_def\_1\_signer.c, [1095](#)
  - tngtls\_cert\_def\_1\_signer.c, [1099](#)
- g\_tngtls\_cert\_elements\_2\_device
  - tnglora\_cert\_def\_2\_device.c, [1096](#)
  - tngtls\_cert\_def\_2\_device.c, [1100](#)
- g\_tngtls\_cert\_elements\_3\_device
  - tngtls\_cert\_def\_3\_device.c, [1102](#)
- g\_tngtls\_cert\_template\_1\_signer
  - tnglora\_cert\_def\_1\_signer.c, [1095](#)
  - tngtls\_cert\_def\_1\_signer.c, [1099](#)
- g\_tngtls\_cert\_template\_2\_device
  - tnglora\_cert\_def\_2\_device.c, [1096](#)
  - tngtls\_cert\_def\_2\_device.c, [1100](#)
- g\_tngtls\_cert\_template\_3\_device
  - tngtls\_cert\_def\_3\_device.c, [1102](#)
- g\_trace\_fp
  - atca\_debug.c, [590](#)
- gen\_dig\_data
  - atca\_temp\_key, [439](#)
- gen\_key\_data
  - atca\_temp\_key, [439](#)
- GENDIG\_COUNT
  - calib\_command.h, [742](#)
- GENDIG\_DATA\_IDX

- calib\_command.h, [742](#)
- GENDIG\_KEYID\_IDX
  - calib\_command.h, [742](#)
- GENDIG\_RSP\_SIZE
  - calib\_command.h, [742](#)
- GENDIG\_ZONE\_CONFIG
  - calib\_command.h, [743](#)
- GENDIG\_ZONE\_COUNTER
  - calib\_command.h, [743](#)
- GENDIG\_ZONE\_DATA
  - calib\_command.h, [743](#)
- GENDIG\_ZONE\_IDX
  - calib\_command.h, [743](#)
- GENDIG\_ZONE\_KEY\_CONFIG
  - calib\_command.h, [743](#)
- GENDIG\_ZONE\_OTP
  - calib\_command.h, [743](#)
- GENDIG\_ZONE\_SHARED\_NONCE
  - calib\_command.h, [744](#)
- GENKEY\_COUNT
  - calib\_command.h, [744](#)
- GENKEY\_COUNT\_DATA
  - calib\_command.h, [744](#)
- GENKEY\_DATA\_IDX
  - calib\_command.h, [744](#)
- GENKEY\_KEYID\_IDX
  - calib\_command.h, [744](#)
- GENKEY\_MODE\_DIGEST
  - calib\_command.h, [744](#)
- GENKEY\_MODE\_IDX
  - calib\_command.h, [745](#)
- GENKEY\_MODE\_MASK
  - calib\_command.h, [745](#)
- GENKEY\_MODE\_PRIVATE
  - calib\_command.h, [745](#)
- GENKEY\_MODE\_PUBKEY\_DIGEST
  - calib\_command.h, [745](#)
- GENKEY\_MODE\_PUBLIC
  - calib\_command.h, [745](#)
- GENKEY\_OTHER\_DATA\_SIZE
  - calib\_command.h, [745](#)
- GENKEY\_PRIVATE\_TO\_TEMPKEY
  - calib\_command.h, [746](#)
- GENKEY\_RSP\_SIZE\_LONG
  - calib\_command.h, [746](#)
- GENKEY\_RSP\_SIZE\_SHORT
  - calib\_command.h, [746](#)

- h
  - atca\_aes\_gcm\_ctx, [412](#)
  - CL\_HashContext, [534](#)
  - hal\_all\_platforms\_kit\_hidapi.c, [833](#)
  - hal\_all\_platforms\_kit\_hidapi.h, [834](#)
  - hal\_check\_wake
    - Hardware abstraction layer (hal\_), [271](#)
  - hal\_create\_mutex
    - Hardware abstraction layer (hal\_), [271](#)
  - hal\_data
    - atca\_iface, [427](#)

ATCAHAL\_t, 456  
 hal\_delay\_10us  
     Hardware abstraction layer (hal\_), 271  
 hal\_delay\_ms  
     Hardware abstraction layer (hal\_), 272  
 hal\_delay\_us  
     Hardware abstraction layer (hal\_), 272  
 hal\_destroy\_mutex  
     Hardware abstraction layer (hal\_), 272  
 hal\_esp32\_i2c.c, 835  
     ACK\_CHECK\_DIS, 836  
     ACK\_CHECK\_EN, 836  
     ACK\_VAL, 836  
     ATCAI2CMaster\_t, 837  
     conf, 846  
     hal\_i2c\_change\_baud, 837  
     hal\_i2c\_discover\_buses, 838  
     hal\_i2c\_discover\_devices, 839  
     hal\_i2c\_idle, 839  
     hal\_i2c\_init, 840  
     hal\_i2c\_post\_init, 842  
     hal\_i2c\_receive, 843  
     hal\_i2c\_release, 844  
     hal\_i2c\_send, 844  
     hal\_i2c\_sleep, 845  
     hal\_i2c\_wake, 846  
     i2c\_bus\_ref\_ct, 846  
     i2c\_hal\_data, 847  
     LOG\_LOCAL\_LEVEL, 836  
     MAX\_I2C\_BUSES, 836  
     NACK\_VAL, 837  
     SCL\_PIN, 837  
     SDA\_PIN, 837  
     TAG, 847  
 hal\_esp32\_timer.c, 847  
     atca\_delay\_ms, 847  
     ets\_delay\_us, 847  
 hal\_free  
     Hardware abstraction layer (hal\_), 272  
 hal\_freertos.c, 848  
     ATCA\_MUTEX\_TIMEOUT, 848  
 hal\_i2c\_change\_baud  
     hal\_esp32\_i2c.c, 837  
 hal\_i2c\_discover\_buses  
     hal\_esp32\_i2c.c, 838  
     Hardware abstraction layer (hal\_), 272  
 hal\_i2c\_discover\_devices  
     hal\_esp32\_i2c.c, 839  
     Hardware abstraction layer (hal\_), 274  
 hal\_i2c\_harmony.c, 848  
 hal\_i2c\_idle  
     hal\_esp32\_i2c.c, 839  
     Hardware abstraction layer (hal\_), 275  
 hal\_i2c\_init  
     hal\_esp32\_i2c.c, 840  
     Hardware abstraction layer (hal\_), 275  
 hal\_i2c\_post\_init  
     hal\_esp32\_i2c.c, 842  
     Hardware abstraction layer (hal\_), 277  
         hal\_esp32\_i2c.c, 843  
         Hardware abstraction layer (hal\_), 277  
 hal\_i2c\_release  
     hal\_esp32\_i2c.c, 844  
     Hardware abstraction layer (hal\_), 278  
 hal\_i2c\_send  
     hal\_esp32\_i2c.c, 844  
     Hardware abstraction layer (hal\_), 279  
 hal\_i2c\_sleep  
     hal\_esp32\_i2c.c, 845  
     Hardware abstraction layer (hal\_), 280  
 hal\_i2c\_start.c, 849  
 hal\_i2c\_start.h, 850  
 hal\_i2c\_wake  
     hal\_esp32\_i2c.c, 846  
     Hardware abstraction layer (hal\_), 280  
 hal\_iface\_init  
     Hardware abstraction layer (hal\_), 281  
 hal\_iface\_register\_hal  
     Hardware abstraction layer (hal\_), 281  
 hal\_iface\_release  
     Hardware abstraction layer (hal\_), 281  
 hal\_kit\_hid\_discover\_buses  
     Hardware abstraction layer (hal\_), 282  
 hal\_kit\_hid\_discover\_devices  
     Hardware abstraction layer (hal\_), 282  
 hal\_kit\_hid\_idle  
     Hardware abstraction layer (hal\_), 283  
 hal\_kit\_hid\_init  
     Hardware abstraction layer (hal\_), 284  
 hal\_kit\_hid\_post\_init  
     Hardware abstraction layer (hal\_), 284  
 hal\_kit\_hid\_receive  
     Hardware abstraction layer (hal\_), 285  
 hal\_kit\_hid\_release  
     Hardware abstraction layer (hal\_), 286  
 hal\_kit\_hid\_send  
     Hardware abstraction layer (hal\_), 286  
 hal\_kit\_hid\_sleep  
     Hardware abstraction layer (hal\_), 287  
 hal\_kit\_hid\_wake  
     Hardware abstraction layer (hal\_), 287  
 hal\_linux.c, 851  
 hal\_linux\_i2c\_userspace.c, 852  
 hal\_linux\_i2c\_userspace.h, 853  
 hal\_linux\_kit\_hid.c, 853  
 hal\_linux\_kit\_hid.h, 854  
 hal\_linux\_spi\_userspace.c, 855  
     hal\_spi\_discover\_buses, 856  
     hal\_spi\_discover\_devices, 856  
     hal\_spi\_idle, 856  
     hal\_spi\_init, 856  
     hal\_spi\_open\_file, 857  
     hal\_spi\_post\_init, 857  
     hal\_spi\_receive, 857  
     hal\_spi\_release, 858

- hal\_spi\_send, [858](#)
- hal\_spi\_sleep, [859](#)
- hal\_spi\_wake, [859](#)
- hal\_linux\_spi\_userspace.h, [860](#)
- hal\_lock\_mutex
  - Hardware abstraction layer (hal\_), [288](#)
- hal\_malloc
  - Hardware abstraction layer (hal\_), [288](#)
- hal\_memset\_s
  - Hardware abstraction layer (hal\_), [263](#)
- hal\_rtos\_delay\_ms
  - Hardware abstraction layer (hal\_), [288](#)
- hal\_sam0\_i2c\_asf.c, [860](#)
- hal\_sam0\_i2c\_asf.h, [861](#)
  - i2c\_sam0\_instance\_t, [862](#)
  - sam0\_change\_baudrate, [862](#)
- hal\_sam\_i2c\_asf.c, [862](#)
- hal\_sam\_i2c\_asf.h, [863](#)
- hal\_sam\_timer\_asf.c, [864](#)
- hal\_spi\_discover\_buses
  - hal\_linux\_spi\_userspace.c, [856](#)
  - Hardware abstraction layer (hal\_), [289](#)
- hal\_spi\_discover\_devices
  - hal\_linux\_spi\_userspace.c, [856](#)
  - Hardware abstraction layer (hal\_), [289](#)
- hal\_spi\_harmony.c, [865](#)
- hal\_spi\_idle
  - hal\_linux\_spi\_userspace.c, [856](#)
  - Hardware abstraction layer (hal\_), [289](#)
- hal\_spi\_init
  - hal\_linux\_spi\_userspace.c, [856](#)
  - Hardware abstraction layer (hal\_), [290](#)
- hal\_spi\_open\_file
  - hal\_linux\_spi\_userspace.c, [857](#)
- hal\_spi\_post\_init
  - hal\_linux\_spi\_userspace.c, [857](#)
  - Hardware abstraction layer (hal\_), [290](#)
- hal\_spi\_receive
  - hal\_linux\_spi\_userspace.c, [857](#)
  - Hardware abstraction layer (hal\_), [290](#)
- hal\_spi\_release
  - hal\_linux\_spi\_userspace.c, [858](#)
  - Hardware abstraction layer (hal\_), [291](#)
- hal\_spi\_send
  - hal\_linux\_spi\_userspace.c, [858](#)
  - Hardware abstraction layer (hal\_), [291](#)
- hal\_spi\_sleep
  - hal\_linux\_spi\_userspace.c, [859](#)
  - Hardware abstraction layer (hal\_), [292](#)
- hal\_spi\_wake
  - hal\_linux\_spi\_userspace.c, [859](#)
  - Hardware abstraction layer (hal\_), [292](#)
- hal\_sw\_i\_discover\_buses
  - Hardware abstraction layer (hal\_), [292](#)
- hal\_sw\_i\_discover\_devices
  - Hardware abstraction layer (hal\_), [293](#)
- hal\_sw\_i\_idle
  - Hardware abstraction layer (hal\_), [293](#)
- hal\_sw\_i\_init
  - Hardware abstraction layer (hal\_), [294](#)
- hal\_sw\_i\_post\_init
  - Hardware abstraction layer (hal\_), [294](#)
- hal\_sw\_i\_receive
  - Hardware abstraction layer (hal\_), [294](#)
- hal\_sw\_i\_release
  - Hardware abstraction layer (hal\_), [295](#)
- hal\_sw\_i\_send
  - Hardware abstraction layer (hal\_), [295](#)
- hal\_sw\_i\_send\_flag
  - Hardware abstraction layer (hal\_), [296](#)
- hal\_sw\_i\_sleep
  - Hardware abstraction layer (hal\_), [296](#)
- hal\_sw\_i\_uart.c, [866](#)
- hal\_sw\_i\_uart.h, [867](#)
- hal\_sw\_i\_wake
  - Hardware abstraction layer (hal\_), [296](#)
- hal\_timer\_start.c, [867](#)
- hal\_uc3\_i2c\_asf.c, [868](#)
- hal\_uc3\_i2c\_asf.h, [869](#)
- hal\_uc3\_timer\_asf.c, [870](#)
- hal\_unlock\_mutex
  - Hardware abstraction layer (hal\_), [297](#)
- hal\_win\_kit\_hid.c, [870](#)
- hal\_win\_kit\_hid.h, [871](#)
- hal\_windows.c, [872](#)
- halidle
  - ATCAHAL\_t, [456](#)
  - ATCAIfaceCfg, [461](#)
- halinit
  - ATCAHAL\_t, [456](#)
  - ATCAIfaceCfg, [462](#)
- halpostinit
  - ATCAHAL\_t, [456](#)
  - ATCAIfaceCfg, [462](#)
- halreceive
  - ATCAHAL\_t, [456](#)
  - ATCAIfaceCfg, [462](#)
- halrelease
  - ATCAHAL\_t, [456](#)
  - ATCAIfaceCfg, [462](#)
- halsend
  - ATCAHAL\_t, [456](#)
  - ATCAIfaceCfg, [462](#)
- halsleep
  - ATCAHAL\_t, [456](#)
  - ATCAIfaceCfg, [462](#)
- halwake
  - ATCAHAL\_t, [457](#)
  - ATCAIfaceCfg, [462](#)
- handle
  - \_pkcs11\_object\_cache\_t, [403](#)
  - \_pkcs11\_session\_ctx, [404](#)
- handle\_info
  - \_pkcs11\_object, [402](#)
  - Hardware abstraction layer (hal\_), [257](#)
  - \_gHid, [307](#)



atca\_delay\_10us, 269  
atca\_delay\_ms, 269  
atca\_delay\_us, 270  
ATCA\_POLLING\_FREQUENCY\_TIME\_MSEC, 262  
ATCA\_POLLING\_INIT\_TIME\_MSEC, 262  
ATCA\_POLLING\_MAX\_TIME\_MSEC, 262  
atcahid\_t, 267  
ATCAI2CMaster\_t, 268  
ATCASPIMaster\_t, 268  
ATCASWIMaster\_t, 268  
change\_i2c\_speed, 270  
DEBUG\_PIN\_1, 262  
DEBUG\_PIN\_2, 263  
hal\_check\_wake, 271  
hal\_create\_mutex, 271  
hal\_delay\_10us, 271  
hal\_delay\_ms, 272  
hal\_delay\_us, 272  
hal\_destroy\_mutex, 272  
hal\_free, 272  
hal\_i2c\_discover\_buses, 272  
hal\_i2c\_discover\_devices, 274  
hal\_i2c\_idle, 275  
hal\_i2c\_init, 275  
hal\_i2c\_post\_init, 277  
hal\_i2c\_receive, 277  
hal\_i2c\_release, 278  
hal\_i2c\_send, 279  
hal\_i2c\_sleep, 280  
hal\_i2c\_wake, 280  
hal\_iface\_init, 281  
hal\_iface\_register\_hal, 281  
hal\_iface\_release, 281  
hal\_kit\_hid\_discover\_buses, 282  
hal\_kit\_hid\_discover\_devices, 282  
hal\_kit\_hid\_idle, 283  
hal\_kit\_hid\_init, 284  
hal\_kit\_hid\_post\_init, 284  
hal\_kit\_hid\_receive, 285  
hal\_kit\_hid\_release, 286  
hal\_kit\_hid\_send, 286  
hal\_kit\_hid\_sleep, 287  
hal\_kit\_hid\_wake, 287  
hal\_lock\_mutex, 288  
hal\_malloc, 288  
hal\_memset\_s, 263  
hal\_rtos\_delay\_ms, 288  
hal\_spi\_discover\_buses, 289  
hal\_spi\_discover\_devices, 289  
hal\_spi\_idle, 289  
hal\_spi\_init, 290  
hal\_spi\_post\_init, 290  
hal\_spi\_receive, 290  
hal\_spi\_release, 291  
hal\_spi\_send, 291  
hal\_spi\_sleep, 292  
hal\_spi\_wake, 292  
hal\_swi\_discover\_buses, 292  
hal\_swi\_discover\_devices, 293  
hal\_swi\_idle, 293  
hal\_swi\_init, 294  
hal\_swi\_post\_init, 294  
hal\_swi\_receive, 294  
hal\_swi\_release, 295  
hal\_swi\_send, 295  
hal\_swi\_send\_flag, 296  
hal\_swi\_sleep, 296  
hal\_swi\_wake, 296  
hal\_unlock\_mutex, 297  
hid\_device\_t, 268  
HID\_DEVICES\_MAX, 263  
HID\_GUID, 263  
HID\_PACKET\_MAX, 263, 264  
i2c\_sam\_instance\_t, 269  
i2c\_start\_instance\_t, 269  
kit\_id\_from\_devtype, 297  
kit\_idle, 297  
kit\_init, 297  
kit\_interface\_from\_kittype, 298  
KIT\_MAX\_SCAN\_COUNT, 264  
KIT\_MAX\_TX\_BUF, 264  
KIT\_MSG\_SIZE, 264  
kit\_parse\_rsp, 298  
kit\_phy\_num\_found, 298  
kit\_phy\_receive, 299, 300  
kit\_phy\_send, 300, 301  
kit\_receive, 301  
KIT\_RX\_WRAP\_SIZE, 264  
kit\_send, 302  
kit\_sleep, 302  
KIT\_TX\_WRAP\_SIZE, 264  
kit\_wake, 303  
kit\_wrap\_cmd, 303  
MAX\_I2C\_BUSES, 264, 265  
MAX\_SPI\_BUSES, 265  
MAX\_SWI\_BUSES, 265  
pin\_conf, 307  
RECEIVE\_MODE, 265  
RX\_DELAY, 266  
sam\_change\_baudrate, 269  
start\_change\_baudrate, 269  
strnchr, 304  
SWI\_FLAG\_CMD, 266  
SWI\_FLAG\_IDLE, 266  
SWI\_FLAG\_SLEEP, 266  
SWI\_FLAG\_TX, 266  
swi\_uart\_deinit, 304  
swi\_uart\_discover\_buses, 304  
swi\_uart\_init, 305  
swi\_uart\_mode, 305  
swi\_uart\_receive\_byte, 306  
swi\_uart\_send\_byte, 306  
swi\_uart\_setbaud, 306  
SWI\_WAKE\_TOKEN, 266  
TRANSMIT\_MODE, 267

- TX\_DELAY, [267](#)
- hardwareVersion
  - CK\_SLOT\_INFO, [511](#)
  - CK\_TOKEN\_INFO, [521](#)
- hash
  - CK\_DSA\_PARAMETER\_GEN\_PARAM, [479](#)
  - sw\_sha256\_ctx, [541](#)
- hashAlg
  - CK\_RSA\_PKCS\_OAEP\_PARAMS, [504](#)
  - CK\_RSA\_PKCS\_PSS\_PARAMS, [505](#)
- hashed\_key
  - atca\_secureboot\_enc\_in\_out, [432](#)
  - atca\_secureboot\_mac\_in\_out, [433](#)
- hClientKey
  - CK\_SSL3\_KEY\_MAT\_OUT, [512](#)
- hClientMacSecret
  - CK\_SSL3\_KEY\_MAT\_OUT, [512](#)
- hid\_device, [534](#)
  - read\_handle, [534](#), [535](#)
  - write\_handle, [535](#)
- hid\_device\_t
  - Hardware abstraction layer (hal\_), [268](#)
- HID\_DEVICES\_MAX
  - Hardware abstraction layer (hal\_), [263](#)
- HID\_GUID
  - Hardware abstraction layer (hal\_), [263](#)
- HID\_PACKET\_MAX
  - Hardware abstraction layer (hal\_), [263](#), [264](#)
- hKey
  - CK\_GOSTR3410\_KEY\_WRAP\_PARAMS, [487](#)
  - CK\_KIP\_PARAMS, [492](#)
  - CK\_WTLS\_KEY\_MAT\_OUT, [524](#)
- HMAC\_COUNT
  - calib\_command.h, [746](#)
- HMAC\_DIGEST\_SIZE
  - calib\_command.h, [746](#)
- HMAC\_KEYID\_IDX
  - calib\_command.h, [746](#)
- HMAC\_MODE\_FLAG\_FULLSN
  - calib\_command.h, [747](#)
- HMAC\_MODE\_FLAG\_OTP64
  - calib\_command.h, [747](#)
- HMAC\_MODE\_FLAG\_OTP88
  - calib\_command.h, [747](#)
- HMAC\_MODE\_FLAG\_TK\_NORAND
  - calib\_command.h, [747](#)
- HMAC\_MODE\_FLAG\_TK\_RAND
  - calib\_command.h, [747](#)
- HMAC\_MODE\_IDX
  - calib\_command.h, [747](#)
- HMAC\_MODE\_MASK
  - calib\_command.h, [748](#)
- HMAC\_RSP\_SIZE
  - calib\_command.h, [748](#)
- hMacSecret
  - CK\_WTLS\_KEY\_MAT\_OUT, [524](#)
- Host side crypto methods (atca\_), [308](#)
  - atca\_check\_mac\_in\_out\_t, [315](#)
  - ATCA\_COMMAND\_HEADER\_SIZE, [312](#)
  - ATCA\_DERIVE\_KEY\_ZEROS\_SIZE, [312](#)
  - atca\_gen\_dig\_in\_out\_t, [315](#)
  - atca\_gen\_key\_in\_out\_t, [315](#)
  - ATCA\_GENDIG\_ZEROS\_SIZE, [312](#)
  - ATCA\_HMAC\_BLOCK\_SIZE, [313](#)
  - atca\_io\_decrypt\_in\_out\_t, [316](#)
  - atca\_mac\_in\_out\_t, [316](#)
  - ATCA\_MSG\_SIZE\_DERIVE\_KEY, [313](#)
  - ATCA\_MSG\_SIZE\_DERIVE\_KEY\_MAC, [313](#)
  - ATCA\_MSG\_SIZE\_ENCRYPT\_MAC, [313](#)
  - ATCA\_MSG\_SIZE\_GEN\_DIG, [313](#)
  - ATCA\_MSG\_SIZE\_HMAC, [313](#)
  - ATCA\_MSG\_SIZE\_MAC, [313](#)
  - ATCA\_MSG\_SIZE\_NONCE, [314](#)
  - ATCA\_MSG\_SIZE\_PRIVWRITE\_MAC, [314](#)
  - atca\_nonce\_in\_out\_t, [316](#)
  - ATCA\_PRIVWRITE\_MAC\_ZEROS\_SIZE, [314](#)
  - ATCA\_PRIVWRITE\_PLAIN\_TEXT\_SIZE, [314](#)
  - atca\_secureboot\_enc\_in\_out\_t, [316](#)
  - atca\_secureboot\_mac\_in\_out\_t, [316](#)
  - atca\_sign\_internal\_in\_out\_t, [316](#)
  - ATCA\_SN\_0\_DEF, [314](#)
  - ATCA\_SN\_1\_DEF, [314](#)
  - ATCA\_SN\_8\_DEF, [314](#)
  - atca\_temp\_key\_t, [316](#)
  - atca\_verify\_in\_out\_t, [317](#)
  - atca\_verify\_mac\_in\_out\_t, [317](#)
  - atca\_write\_mac\_in\_out\_t, [317](#)
  - ATCA\_WRITE\_MAC\_ZEROS\_SIZE, [315](#)
  - atcah\_check\_mac, [317](#)
  - atcah\_config\_to\_sign\_internal, [317](#)
  - atcah\_decrypt, [318](#)
  - atcah\_derive\_key, [318](#)
  - atcah\_derive\_key\_mac, [319](#)
  - atcah\_encode\_counter\_match, [319](#)
  - atcah\_gen\_dig, [320](#)
  - atcah\_gen\_key\_msg, [320](#)
  - atcah\_gen\_mac, [321](#)
  - atcah\_hmac, [321](#)
  - atcah\_include\_data, [321](#)
  - atcah\_io\_decrypt, [322](#)
  - atcah\_mac, [322](#)
  - atcah\_nonce, [323](#)
  - atcah\_privwrite\_auth\_mac, [323](#)
  - atcah\_secureboot\_enc, [323](#)
  - atcah\_secureboot\_mac, [324](#)
  - atcah\_sha256, [324](#)
  - atcah\_sign\_internal\_msg, [324](#)
  - atcah\_verify\_mac, [325](#)
  - atcah\_write\_auth\_mac, [325](#)
  - challenge, [326](#)
  - crypto\_data, [326](#)
  - curve\_type, [326](#)
  - ENCRYPTION\_KEY\_SIZE, [315](#)
  - key, [326](#)
  - key\_id, [327](#)
  - MAC\_MODE\_USE\_TEMPKEY\_MASK, [315](#)



- mode, [327](#)
- num\_in, [327](#)
- otp, [328](#)
- p\_temp, [328](#)
- public\_key, [328](#)
- rand\_out, [328](#)
- response, [329](#)
- signature, [329](#)
- sn, [329](#)
- temp\_key, [330](#)
- zero, [330](#)
- host\_generate\_random\_number
  - secure\_boot.h, [1067](#)
- hPrivateKeyData
  - CK\_ECDH2\_DERIVE\_PARAMS, [481](#)
  - CK\_ECMQV\_DERIVE\_PARAMS, [483](#)
  - CK\_X9\_42\_DH2\_DERIVE\_PARAMS, [530](#)
  - CK\_X9\_42\_MQV\_DERIVE\_PARAMS, [532](#)
- hServerKey
  - CK\_SSL3\_KEY\_MAT\_OUT, [512](#)
- hServerMacSecret
  - CK\_SSL3\_KEY\_MAT\_OUT, [513](#)
- hw\_sha256\_ctx, [535](#)
  - block, [535](#)
  - block\_size, [535](#)
  - total\_msg\_size, [536](#)
- I2C\_Address
  - \_atecc508a\_config, [388](#)
  - \_atecc608\_config, [392](#)
  - \_atsha204a\_config, [396](#)
- i2c\_bus\_ref\_ct
  - hal\_esp32\_i2c.c, [846](#)
- i2c\_descriptor
  - i2c\_start\_instance, [537](#)
- I2C\_Enable
  - \_atecc508a\_config, [388](#)
  - \_atecc608\_config, [392](#)
  - \_atsha204a\_config, [396](#)
- i2c\_file
  - atcal2Cmaster, [458](#)
- i2c\_hal\_data
  - hal\_esp32\_i2c.c, [847](#)
- i2c\_instance
  - i2c\_sam0\_instance, [536](#)
  - i2c\_sam\_instance, [537](#)
- i2c\_sam0\_instance, [536](#)
  - change\_baudrate, [536](#)
  - i2c\_instance, [536](#)
- i2c\_sam0\_instance\_t
  - hal\_sam0\_i2c\_asf.h, [862](#)
- i2c\_sam\_instance, [536](#)
  - change\_baudrate, [537](#)
  - i2c\_instance, [537](#)
- i2c\_sam\_instance\_t
  - Hardware abstraction layer (hal\_), [269](#)
- i2c\_start\_instance, [537](#)
  - change\_baudrate, [537](#)
  - i2c\_descriptor, [537](#)
- i2c\_start\_instance\_t
  - Hardware abstraction layer (hal\_), [269](#)
- id
  - atcacert\_cert\_element\_s, [447](#)
  - atcal2Cmaster, [458](#)
- idx
  - ATCAIfaceCfg, [462](#)
- iface\_type
  - ATCAIfaceCfg, [463](#)
- INCIDENTAL
  - license.txt, [887](#)
- INCLUDING
  - license.txt, [887](#)
- INDIRECT
  - license.txt, [888](#)
- info
  - \_pcks11\_mech\_table\_e, [398](#)
- INFO\_COUNT
  - calib\_command.h, [748](#)
- INFO\_DRIVER\_STATE\_MASK
  - calib\_command.h, [748](#)
- INFO\_MODE\_GPIO
  - calib\_command.h, [748](#)
- INFO\_MODE\_KEY\_VALID
  - calib\_command.h, [748](#)
- INFO\_MODE\_MAX
  - calib\_command.h, [749](#)
- INFO\_MODE\_REVISION
  - calib\_command.h, [749](#)
- INFO\_MODE\_STATE
  - calib\_command.h, [749](#)
- INFO\_MODE\_VOL\_KEY\_PERMIT
  - calib\_command.h, [749](#)
- INFO\_NO\_STATE
  - calib\_command.h, [749](#)
- INFO\_OUTPUT\_STATE\_MASK
  - calib\_command.h, [749](#)
- INFO\_PARAM1\_IDX
  - calib\_command.h, [750](#)
- INFO\_PARAM2\_IDX
  - calib\_command.h, [750](#)
- INFO\_PARAM2\_LATCH\_CLEAR
  - calib\_command.h, [750](#)
- INFO\_PARAM2\_LATCH\_SET
  - calib\_command.h, [750](#)
- INFO\_PARAM2\_SET\_LATCH\_STATE
  - calib\_command.h, [750](#)
- INFO\_RSP\_SIZE
  - calib\_command.h, [750](#)
- INFO\_SIZE
  - calib\_command.h, [751](#)
- INFRINGEMENT
  - license.txt, [888](#)
- initATCACCommand
  - ATCACCommand (atca\_), [123](#)
- initATCADevice
  - ATCADevice (atca\_), [144](#)
- initATCAIface

- ATCAIface (atca\_), 151
- initialized
  - \_pkcs11\_lib\_ctx, 399
  - \_pkcs11\_session\_ctx, 404
  - \_pkcs11\_slot\_ctx, 406
- input\_data
  - atca\_write\_mac\_in\_out, 444
- interface\_config
  - \_pkcs11\_slot\_ctx, 406
- io\_key
  - atca\_io\_decrypt\_in\_out, 429
  - atca\_secureboot\_enc\_in\_out, 432
  - atca\_verify\_mac, 442
- io\_protection\_get\_key
  - io\_protection\_key.h, 873
- io\_protection\_key.h, 873
  - io\_protection\_get\_key, 873
  - io\_protection\_set\_key, 873
- io\_protection\_set\_key
  - io\_protection\_key.h, 873
- is\_64
  - atca\_temp\_key, 439
- is\_device\_sn
  - atcacert\_build\_state\_s, 446
- is\_genkey
  - atcacert\_device\_loc\_s, 453
- is\_key\_nomac
  - atca\_gen\_dig\_in\_out, 422
- is\_slot\_locked
  - atca\_sign\_internal\_in\_out, 437
- isAlpha
  - atca\_helpers.c, 605
  - atca\_helpers.h, 616
- isATCAError
  - calib\_command.c, 698
  - calib\_command.h, 800
- isBase64
  - atca\_helpers.c, 605
  - atca\_helpers.h, 616
- isBase64Digit
  - atca\_helpers.c, 606
  - atca\_helpers.h, 616
- isDigit
  - atca\_helpers.c, 606
  - atca\_helpers.h, 617
- isHex
  - atca\_helpers.c, 607
  - atca\_helpers.h, 617
- isHexAlpha
  - atca\_helpers.c, 607
  - atca\_helpers.h, 617
- isHexDigit
  - atca\_helpers.c, 607
  - atca\_helpers.h, 618
- isSender
  - CK\_KEA\_DERIVE\_PARAMS, 489
- issue\_date\_format
  - atcacert\_def\_s, 451

- isWhiteSpace
  - atca\_helpers.c, 608
  - atca\_helpers.h, 618
- iterations
  - CK\_PKCS5\_PBKD2\_PARAMS, 497
  - CK\_PKCS5\_PBKD2\_PARAMS2, 499
- iv
  - CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS, 467
  - CK\_ARIA\_CBC\_ENCRYPT\_DATA\_PARAMS, 471
  - CK\_CAMELLIA\_CBC\_ENCRYPT\_DATA\_PARAMS, 474
  - CK\_DES\_CBC\_ENCRYPT\_DATA\_PARAMS, 478
  - CK\_RC2\_CBC\_PARAMS, 500
  - CK\_SEED\_CBC\_ENCRYPT\_DATA\_PARAMS, 505
- j0
  - atca\_aes\_gcm\_ctx, 412
- JSON Web Token (JWT) methods (atca\_jwt\_), 331
  - atca\_jwt\_add\_claim\_numeric, 331
  - atca\_jwt\_add\_claim\_string, 331
  - atca\_jwt\_check\_payload\_start, 332
  - atca\_jwt\_finalize, 332
  - atca\_jwt\_init, 332
  - atca\_jwt\_verify, 332
- kdf
  - CK\_ECDH1\_DERIVE\_PARAMS, 480
  - CK\_ECDH2\_DERIVE\_PARAMS, 481
  - CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS, 482
  - CK\_ECMQV\_DERIVE\_PARAMS, 483
  - CK\_GOSTR3410\_DERIVE\_PARAMS, 486
  - CK\_X9\_42\_DH1\_DERIVE\_PARAMS, 529
  - CK\_X9\_42\_DH2\_DERIVE\_PARAMS, 530
  - CK\_X9\_42\_MQV\_DERIVE\_PARAMS, 532
  - KDF\_DETAILS\_AES\_KEY\_LOC\_MASK
    - calib\_command.h, 751
  - KDF\_DETAILS\_HKDF\_MSG\_LOC\_INPUT
    - calib\_command.h, 751
  - KDF\_DETAILS\_HKDF\_MSG\_LOC\_IV
    - calib\_command.h, 751
  - KDF\_DETAILS\_HKDF\_MSG\_LOC\_MASK
    - calib\_command.h, 751
  - KDF\_DETAILS\_HKDF\_MSG\_LOC\_SLOT
    - calib\_command.h, 751
  - KDF\_DETAILS\_HKDF\_MSG\_LOC\_TEMPKEY
    - calib\_command.h, 752
  - KDF\_DETAILS\_HKDF\_ZERO\_KEY
    - calib\_command.h, 752
  - KDF\_DETAILS\_IDX
    - calib\_command.h, 752
  - KDF\_DETAILS\_PRF\_AEAD\_MASK
    - calib\_command.h, 752
  - KDF\_DETAILS\_PRF\_AEAD\_MODE0
    - calib\_command.h, 752
  - KDF\_DETAILS\_PRF\_AEAD\_MODE1
    - calib\_command.h, 752
  - KDF\_DETAILS\_PRF\_KEY\_LEN\_16
    - calib\_command.h, 753
  - KDF\_DETAILS\_PRF\_KEY\_LEN\_32

- calib\_command.h, [753](#)
- KDF\_DETAILS\_PRF\_KEY\_LEN\_48
  - calib\_command.h, [753](#)
- KDF\_DETAILS\_PRF\_KEY\_LEN\_64
  - calib\_command.h, [753](#)
- KDF\_DETAILS\_PRF\_KEY\_LEN\_MASK
  - calib\_command.h, [753](#)
- KDF\_DETAILS\_PRF\_TARGET\_LEN\_32
  - calib\_command.h, [753](#)
- KDF\_DETAILS\_PRF\_TARGET\_LEN\_64
  - calib\_command.h, [754](#)
- KDF\_DETAILS\_PRF\_TARGET\_LEN\_MASK
  - calib\_command.h, [754](#)
- KDF\_DETAILS\_SIZE
  - calib\_command.h, [754](#)
- KDF\_KEYID\_IDX
  - calib\_command.h, [754](#)
- KDF\_MESSAGE\_IDX
  - calib\_command.h, [754](#)
- KDF\_MODE\_ALG\_AES
  - calib\_command.h, [754](#)
- KDF\_MODE\_ALG\_HKDF
  - calib\_command.h, [755](#)
- KDF\_MODE\_ALG\_MASK
  - calib\_command.h, [755](#)
- KDF\_MODE\_ALG\_PRF
  - calib\_command.h, [755](#)
- KDF\_MODE\_IDX
  - calib\_command.h, [755](#)
- KDF\_MODE\_SOURCE\_ALTKEYBUF
  - calib\_command.h, [755](#)
- KDF\_MODE\_SOURCE\_MASK
  - calib\_command.h, [755](#)
- KDF\_MODE\_SOURCE\_SLOT
  - calib\_command.h, [756](#)
- KDF\_MODE\_SOURCE\_TEMPKEY
  - calib\_command.h, [756](#)
- KDF\_MODE\_SOURCE\_TEMPKEY\_UP
  - calib\_command.h, [756](#)
- KDF\_MODE\_TARGET\_ALTKEYBUF
  - calib\_command.h, [756](#)
- KDF\_MODE\_TARGET\_MASK
  - calib\_command.h, [756](#)
- KDF\_MODE\_TARGET\_OUTPUT
  - calib\_command.h, [756](#)
- KDF\_MODE\_TARGET\_OUTPUT\_ENC
  - calib\_command.h, [757](#)
- KDF\_MODE\_TARGET\_SLOT
  - calib\_command.h, [757](#)
- KDF\_MODE\_TARGET\_TEMPKEY
  - calib\_command.h, [757](#)
- KDF\_MODE\_TARGET\_TEMPKEY\_UP
  - calib\_command.h, [757](#)
- KdfivLoc
  - \_atecc608\_config, [392](#)
- KdfivStr
  - \_atecc608\_config, [392](#)
- key
  - Host side crypto methods (atcah\_), [326](#)
- key\_block
  - atca\_aes\_cbc\_ctx, [408](#)
  - atca\_aes\_ctr\_ctx, [410](#)
  - atca\_aes\_gcm\_ctx, [412](#)
- key\_conf
  - atca\_gen\_dig\_in\_out, [422](#)
- key\_config
  - atca\_sign\_internal\_in\_out, [437](#)
- key\_id
  - atca\_aes\_cbc\_ctx, [408](#)
  - atca\_aes\_ctr\_ctx, [410](#)
  - atca\_aes\_gcm\_ctx, [412](#)
  - atca\_check\_mac\_in\_out, [414](#)
  - atca\_gen\_dig\_in\_out, [422](#)
  - atca\_gen\_key\_in\_out, [424](#)
  - atca\_sign\_internal\_in\_out, [437](#)
  - atca\_temp\_key, [440](#)
  - atca\_verify\_mac, [442](#)
  - atca\_write\_mac\_in\_out, [444](#)
  - Host side crypto methods (atcah\_), [327](#)
- KeyConfig
  - \_atecc508a\_config, [388](#)
  - \_atecc608\_config, [392](#)
- kit\_id\_from\_devtype
  - Hardware abstraction layer (hal\_), [297](#)
- kit\_idle
  - Hardware abstraction layer (hal\_), [297](#)
- kit\_init
  - Hardware abstraction layer (hal\_), [297](#)
- kit\_interface\_from\_kittype
  - Hardware abstraction layer (hal\_), [298](#)
- KIT\_MAX\_SCAN\_COUNT
  - Hardware abstraction layer (hal\_), [264](#)
- KIT\_MAX\_TX\_BUF
  - Hardware abstraction layer (hal\_), [264](#)
- KIT\_MSG\_SIZE
  - Hardware abstraction layer (hal\_), [264](#)
- kit\_parse\_rsp
  - Hardware abstraction layer (hal\_), [298](#)
- kit\_phy.h, [874](#)
- kit\_phy\_num\_found
  - Hardware abstraction layer (hal\_), [298](#)
- kit\_phy\_receive
  - Hardware abstraction layer (hal\_), [299, 300](#)
- kit\_phy\_send
  - Hardware abstraction layer (hal\_), [300, 301](#)
- kit\_protocol.c, [874](#)
- kit\_protocol.h, [875](#)
- kit\_receive
  - Hardware abstraction layer (hal\_), [301](#)
- KIT\_RX\_WRAP\_SIZE
  - Hardware abstraction layer (hal\_), [264](#)
- kit\_send
  - Hardware abstraction layer (hal\_), [302](#)
- kit\_sleep
  - Hardware abstraction layer (hal\_), [302](#)
- KIT\_TX\_WRAP\_SIZE

- Hardware abstraction layer (hal\_), 264
- kit\_wake
  - Hardware abstraction layer (hal\_), 303
- kit\_wrap\_cmd
  - Hardware abstraction layer (hal\_), 303
- kits
  - atcahid, 457
- label
  - \_pkcs11\_slot\_ctx, 406
  - CK\_TOKEN\_INFO, 521
- LastKeyUse
  - \_atecc508a\_config, 388
  - \_atsha204a\_config, 396
- LAW
  - license.txt, 888
- leftRotate
  - sha1\_routines.h, 1072
- length
  - CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS, 468
  - CK\_ARIA\_CBC\_ENCRYPT\_DATA\_PARAMS, 471
  - CK\_CAMELLIA\_CBC\_ENCRYPT\_DATA\_PARAMS, 474
  - CK\_DES\_CBC\_ENCRYPT\_DATA\_PARAMS, 478
  - CK\_SEED\_CBC\_ENCRYPT\_DATA\_PARAMS, 506
- LIABILITY
  - license.txt, 889
- libraryDescription
  - CK\_INFO, 489
- libraryVersion
  - CK\_INFO, 489
- License
  - license.txt, 890
- license
  - license.txt, 889
- license.txt, 876
  - ANY, 884
  - CAUSED, 884
  - CONTRACT, 884
  - DAMAGE, 885
  - DAMAGES, 882
  - DIRECT, 885
  - EXEMPLARY, 886
  - EXPRESS, 886
  - FEES, 886
  - forms, 886
  - Foundation, 887
  - INCIDENTAL, 887
  - INCLUDING, 887
  - INDIRECT, 888
  - INFRINGEMENT, 888
  - LAW, 888
  - LIABILITY, 889
  - License, 890
  - license, 889
  - LOSS, 890
  - MERCHANTABILITY, 890
  - met, 890
  - modification, 890
  - not, 891
  - notice, 891
  - or, 883
  - Ott, 891
  - PUNITIVE, 891
  - SOFTWARE, 892
  - software, 883
  - SPECIAL, 892
  - STATUTORY, 892
  - systemd, 893
  - terms, 893
  - TO, 893
  - TORT, 883
  - WARRANTIES, 893
  - WARRANTY, 894
- LOCK\_COUNT
  - calib\_command.h, 757
- lock\_mutex
  - \_pkcs11\_lib\_ctx, 400
- LOCK\_RSP\_SIZE
  - calib\_command.h, 757
- LOCK\_SUMMARY\_IDX
  - calib\_command.h, 758
- LOCK\_ZONE\_CONFIG
  - calib\_command.h, 758
- LOCK\_ZONE\_DATA
  - calib\_command.h, 758
- LOCK\_ZONE\_DATA\_SLOT
  - calib\_command.h, 758
- LOCK\_ZONE\_IDX
  - calib\_command.h, 758
- LOCK\_ZONE\_MASK
  - calib\_command.h, 758
- LOCK\_ZONE\_NO\_CRC
  - calib\_command.h, 759
- LockConfig
  - \_atecc508a\_config, 388
  - \_atecc608\_config, 393
  - \_atsha204a\_config, 396
- LockMutex
  - CK\_C\_INITIALIZE\_ARGS, 473
- LockValue
  - \_atecc508a\_config, 388
  - \_atecc608\_config, 393
  - \_atsha204a\_config, 396
- LOG\_LOCAL\_LEVEL
  - hal\_esp32\_i2c.c, 836
- logged\_in
  - \_pkcs11\_session\_ctx, 404
- LOSS
  - license.txt, 890
- mac
  - atca\_derive\_key\_mac\_in\_out, 419
  - atca\_secureboot\_mac\_in\_out, 434
  - atca\_verify\_mac, 442
- MAC\_CHALLENGE\_IDX
  - calib\_command.h, 759
- MAC\_CHALLENGE\_SIZE

- calib\_command.h, 759
- MAC\_COUNT\_LONG
  - calib\_command.h, 759
- MAC\_COUNT\_SHORT
  - calib\_command.h, 759
- MAC\_KEYID\_IDX
  - calib\_command.h, 759
- MAC\_MODE\_BLOCK1\_TEMPKEY
  - calib\_command.h, 760
- MAC\_MODE\_BLOCK2\_TEMPKEY
  - calib\_command.h, 760
- MAC\_MODE\_CHALLENGE
  - calib\_command.h, 760
- MAC\_MODE\_IDX
  - calib\_command.h, 760
- MAC\_MODE\_INCLUDE\_OTP\_64
  - calib\_command.h, 760
- MAC\_MODE\_INCLUDE\_OTP\_88
  - calib\_command.h, 760
- MAC\_MODE\_INCLUDE\_SN
  - calib\_command.h, 761
- MAC\_MODE\_MASK
  - calib\_command.h, 761
- MAC\_MODE\_PASSTHROUGH
  - calib\_command.h, 761
- MAC\_MODE\_PTNONCE\_TEMPKEY
  - calib\_command.h, 761
- MAC\_MODE\_SOURCE\_FLAG\_MATCH
  - calib\_command.h, 761
- MAC\_MODE\_USE\_TEMPKEY\_MASK
  - Host side crypto methods (atcah\_), 315
- MAC\_RSP\_SIZE
  - calib\_command.h, 761
- MAC\_SIZE
  - calib\_command.h, 762
- major
  - CK\_VERSION, 524
- manufacturerID
  - CK\_INFO, 489
  - CK\_SLOT\_INFO, 512
  - CK\_TOKEN\_INFO, 522
- MAX\_BUSES
  - calib\_basic.c, 679
- max\_cert\_size
  - atcacert\_build\_state\_s, 446
- MAX\_I2C\_BUSES
  - hal\_esp32\_i2c.c, 836
  - Hardware abstraction layer (hal\_), 264, 265
- MAX\_SPI\_BUSES
  - Hardware abstraction layer (hal\_), 265
- MAX\_SWI\_BUSES
  - Hardware abstraction layer (hal\_), 265
- mbedTLS Wrapper methods (atca\_mbedtls\_), 333
  - atca\_mbedtls\_cert\_add, 333
  - atca\_mbedtls\_ecdh\_ioprot\_cb, 333
  - atca\_mbedtls\_ecdh\_slot\_cb, 333
  - atca\_mbedtls\_pk\_init, 334
- mbedtls\_calloc
  - atca\_mbedtls\_wrap.c, 630
- MBEDTLS\_CMAC\_C
  - atca\_crypto\_sw.h, 581
- mbedtls\_free
  - atca\_mbedtls\_wrap.c, 630
- mCommands
  - atca\_device, 420
- mechanism
  - CK\_MECHANISM, 493
- memcpy\_P
  - sha1\_routines.h, 1072
- memory\_parameters, 538
  - memory\_size, 538
  - reserved, 538
  - signature, 538
  - start\_address, 538
  - version\_info, 538
- memory\_params
  - secure\_boot\_parameters, 540
- memory\_size
  - memory\_parameters, 538
- MERCHANTABILITY
  - license.txt, 890
- message
  - atca\_sign\_internal\_in\_out, 437
- met
  - license.txt, 890
- mgf
  - CK\_RSA\_PKCS\_OAEP\_PARAMS, 504
  - CK\_RSA\_PKCS\_PSS\_PARAMS, 505
- mlface
  - atca\_device, 420
- mlfaceCFG
  - atca\_iface, 427
- minor
  - CK\_VERSION, 524
- mode
  - atca\_check\_mac\_in\_out, 414
  - atca\_derive\_key\_in\_out, 417
  - atca\_derive\_key\_mac\_in\_out, 419
  - atca\_gen\_key\_in\_out, 424
  - atca\_include\_data\_in\_out, 428
  - atca\_secureboot\_mac\_in\_out, 434
  - atca\_sign\_internal\_in\_out, 437
  - atca\_verify\_mac, 442
  - Host side crypto methods (atcah\_), 327
- model
  - CK\_TOKEN\_INFO, 522
- modification
  - license.txt, 890
- month
  - CK\_DATE, 477
- msg\_dig\_buf
  - atca\_verify\_mac, 442
- mType
  - atca\_iface, 428
- mutex
  - \_pkcs11\_lib\_ctx, 400

NACK\_VAL  
     hal\_esp32\_i2c.c, [837](#)  
 name  
     \_pkcs11\_object, [402](#)  
 newATCACCommand  
     ATCACCommand (atca\_), [123](#)  
 newATCADevice  
     ATCADevice (atca\_), [144](#)  
 newATCAIface  
     ATCAIface (atca\_), [151](#)  
 no\_mac\_flag  
     atca\_temp\_key, [440](#)  
 NONCE\_COUNT\_LONG  
     calib\_command.h, [762](#)  
 NONCE\_COUNT\_LONG\_64  
     calib\_command.h, [762](#)  
 NONCE\_COUNT\_SHORT  
     calib\_command.h, [762](#)  
 NONCE\_INPUT\_IDX  
     calib\_command.h, [762](#)  
 NONCE\_MODE\_IDX  
     calib\_command.h, [762](#)  
 NONCE\_MODE\_INPUT\_LEN\_32  
     calib\_command.h, [763](#)  
 NONCE\_MODE\_INPUT\_LEN\_64  
     calib\_command.h, [763](#)  
 NONCE\_MODE\_INPUT\_LEN\_MASK  
     calib\_command.h, [763](#)  
 NONCE\_MODE\_INVALID  
     calib\_command.h, [763](#)  
 NONCE\_MODE\_MASK  
     calib\_command.h, [763](#)  
 NONCE\_MODE\_NO\_SEED\_UPDATE  
     calib\_command.h, [763](#)  
 NONCE\_MODE\_PASSTHROUGH  
     calib\_command.h, [764](#)  
 NONCE\_MODE\_SEED\_UPDATE  
     calib\_command.h, [764](#)  
 NONCE\_MODE\_TARGET\_ALTKEYBUF  
     calib\_command.h, [764](#)  
 NONCE\_MODE\_TARGET\_MASK  
     calib\_command.h, [764](#)  
 NONCE\_MODE\_TARGET\_MSGDIGBUF  
     calib\_command.h, [764](#)  
 NONCE\_MODE\_TARGET\_TEMPKEY  
     calib\_command.h, [764](#)  
 NONCE\_NUMIN\_SIZE  
     calib\_command.h, [765](#)  
 NONCE\_NUMIN\_SIZE\_PASSTHROUGH  
     calib\_command.h, [765](#)  
 NONCE\_PARAM2\_IDX  
     calib\_command.h, [765](#)  
 NONCE\_RSP\_SIZE\_LONG  
     calib\_command.h, [765](#)  
 NONCE\_RSP\_SIZE\_SHORT  
     calib\_command.h, [765](#)  
 NONCE\_ZERO\_CALC\_MASK  
     calib\_command.h, [765](#)  
 NONCE\_ZERO\_CALC\_RANDOM  
     calib\_command.h, [766](#)  
 NONCE\_ZERO\_CALC\_TEMPKEY  
     calib\_command.h, [766](#)  
 not  
     license.txt, [891](#)  
 notice  
     license.txt, [891](#)  
 NULL\_PTR  
     cryptoki.h, [827](#)  
 num\_in  
     Host side crypto methods (atcah\_), [327](#)  
 num\_kits\_found  
     atcahid, [457](#)  
  
 object  
     \_pkcs11\_object\_cache\_t, [403](#)  
 object\_count  
     \_pkcs11\_session\_ctx, [404](#)  
 object\_index  
     \_pkcs11\_session\_ctx, [404](#)  
 offset  
     atcacert\_cert\_loc\_s, [448](#)  
     atcacert\_device\_loc\_s, [453](#)  
 opcode  
     ATCAPacket, [465](#)  
 or  
     license.txt, [883](#)  
 other\_data  
     atca\_check\_mac\_in\_out, [414](#)  
     atca\_gen\_dig\_in\_out, [422](#)  
     atca\_gen\_key\_in\_out, [424](#)  
     atca\_verify\_mac, [442](#)  
 otp  
     atca\_check\_mac\_in\_out, [414](#)  
     Host side crypto methods (atcah\_), [328](#)  
 otpcode  
     tng\_cert\_map\_element, [542](#)  
 OTPmode  
     \_atecc508a\_config, [389](#)  
     \_atsha204a\_config, [396](#)  
 Ott  
     license.txt, [891](#)  
 out\_nonce  
     atca\_io\_decrypt\_in\_out, [429](#)  
 OUTNONCE\_SIZE  
     calib\_command.h, [766](#)  
  
 p\_temp  
     Host side crypto methods (atcah\_), [328](#)  
 pAAD  
     CK\_AES\_CCM\_PARAMS, [468](#)  
     CK\_AES\_GCM\_PARAMS, [470](#)  
     CK\_CCM\_PARAMS, [475](#)  
     CK\_GCM\_PARAMS, [485](#)  
 packetsize  
     ATCAIfaceCfg, [463](#)  
 packHex  
     atca\_helpers.c, [608](#)

---

- atca\_helpers.h, [619](#)
- pApplication
  - pkcs11t.h, [1062](#)
- param1
  - ATCAPacket, [465](#)
- param2
  - atca\_secureboot\_mac\_in\_out, [434](#)
  - ATCAPacket, [465](#)
- parent\_key
  - atca\_derive\_key\_in\_out, [417](#)
  - atca\_derive\_key\_mac\_in\_out, [419](#)
- parity
  - ATCAIfaceCfg, [463](#)
- partial\_aad
  - atca\_aes\_gcm\_ctx, [413](#)
- partial\_aad\_size
  - atca\_aes\_gcm\_ctx, [413](#)
- PAUSE\_COUNT
  - calib\_command.h, [766](#)
- PAUSE\_PARAM2\_IDX
  - calib\_command.h, [766](#)
- PAUSE\_RSP\_SIZE
  - calib\_command.h, [766](#)
- PAUSE\_SELECT\_IDX
  - calib\_command.h, [767](#)
- pBaseG
  - CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS, [507](#)
- PKCS11\_MECH\_ECC508\_EC\_CAPABILITY
  - Attributes (pkcs11\_attrib\_), [342](#)
- pkcs11\_mech\_table\_e
  - Attributes (pkcs11\_attrib\_), [342](#)
- pkcs11\_mech\_table\_ptr
  - Attributes (pkcs11\_attrib\_), [342](#)
- pClientRandom
  - CK\_SSL3\_RANDOM\_DATA, [515](#)
  - CK\_WTLS\_RANDOM\_DATA, [528](#)
- pContentType
  - CK\_CMS\_SIG\_PARAMS, [476](#)
- pContextData
  - CK\_TLS\_KDF\_PARAMS, [518](#)
- pData
  - CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS, [468](#)
  - CK\_ARIA\_CBC\_ENCRYPT\_DATA\_PARAMS, [471](#)
  - CK\_CAMELLIA\_CBC\_ENCRYPT\_DATA\_PARAMS, [474](#)
  - CK\_DES\_CBC\_ENCRYPT\_DATA\_PARAMS, [478](#)
  - CK\_KEY\_DERIVATION\_STRING\_DATA, [490](#)
  - CK\_SEED\_CBC\_ENCRYPT\_DATA\_PARAMS, [506](#)
- pDigestMechanism
  - CK\_CMS\_SIG\_PARAMS, [476](#)
- PEM\_CERT\_BEGIN
  - atcacert\_pem.h, [668](#)
- PEM\_CERT\_END
  - atcacert\_pem.h, [668](#)
- PEM\_CSR\_BEGIN
  - atcacert\_pem.h, [668](#)
- PEM\_CSR\_END
  - atcacert\_pem.h, [669](#)
- pid
  - ATCAIfaceCfg, [463](#)
- pin\_conf
  - Hardware abstraction layer (hal\_), [307](#)
- plnitVector
  - CK\_PBE\_PARAMS, [496](#)
- plV
  - CK\_WTLS\_KEY\_MAT\_OUT, [524](#)
- plv
  - CK\_AES\_GCM\_PARAMS, [470](#)
  - CK\_GCM\_PARAMS, [485](#)
  - CK\_RC5\_CBC\_PARAMS, [501](#)
- plVClient
  - CK\_SSL3\_KEY\_MAT\_OUT, [513](#)
- plVServer
  - CK\_SSL3\_KEY\_MAT\_OUT, [513](#)
- pkcs11.h, [894](#)
  - \_\_PASTE, [894](#)
  - CK\_NEED\_ARG\_LIST, [895](#)
  - CK\_PKCS11\_FUNCTION\_INFO, [895](#)
- PKCS11\_API
  - cryptoki.h, [828](#)
- pkcs11\_attrib.c, [895](#)
- pkcs11\_attrib.h, [896](#)
  - attrib\_f, [897](#)
  - pkcs11\_attrib\_model, [897](#)
  - pkcs11\_attrib\_model\_ptr, [897](#)
- pkcs11\_attrib\_empty
  - Attributes (pkcs11\_attrib\_), [358](#)
- pkcs11\_attrib\_false
  - Attributes (pkcs11\_attrib\_), [358](#)
- pkcs11\_attrib\_fill
  - Attributes (pkcs11\_attrib\_), [358](#)
- pkcs11\_attrib\_model
  - pkcs11\_attrib.h, [897](#)
- pkcs11\_attrib\_model\_ptr
  - pkcs11\_attrib.h, [897](#)
- pkcs11\_attrib\_true
  - Attributes (pkcs11\_attrib\_), [358](#)
- pkcs11\_attrib\_value
  - Attributes (pkcs11\_attrib\_), [359](#)
- pkcs11\_cert.c, [897](#)
- pkcs11\_cert.h, [898](#)
- pkcs11\_cert\_get\_authority\_key\_id
  - Attributes (pkcs11\_attrib\_), [359](#)
- pkcs11\_cert\_get\_encoded
  - Attributes (pkcs11\_attrib\_), [359](#)
- pkcs11\_cert\_get\_subject
  - Attributes (pkcs11\_attrib\_), [359](#)
- pkcs11\_cert\_get\_subject\_key\_id
  - Attributes (pkcs11\_attrib\_), [359](#)
- pkcs11\_cert\_get\_trusted\_flag
  - Attributes (pkcs11\_attrib\_), [359](#)
- pkcs11\_cert\_get\_type
  - Attributes (pkcs11\_attrib\_), [360](#)
- pkcs11\_cert\_wtlspublic\_attributes
  - Attributes (pkcs11\_attrib\_), [374](#)
- pkcs11\_cert\_wtlspublic\_attributes\_count



- Attributes (pkcs11\_attrib\_), 374
- pkcs11\_cert\_x509\_attributes
  - Attributes (pkcs11\_attrib\_), 374
- pkcs11\_cert\_x509\_attributes\_count
  - Attributes (pkcs11\_attrib\_), 374
- pkcs11\_cert\_x509\_write
  - Attributes (pkcs11\_attrib\_), 360
- pkcs11\_cert\_x509public\_attributes
  - Attributes (pkcs11\_attrib\_), 374
- pkcs11\_cert\_x509public\_attributes\_count
  - Attributes (pkcs11\_attrib\_), 374
- pkcs11\_config.c, 899
- pkcs11\_config\_cert
  - Attributes (pkcs11\_attrib\_), 360
  - example\_pkcs11\_config.c, 832
- pkcs11\_config\_init\_cert
  - Attributes (pkcs11\_attrib\_), 360
- pkcs11\_config\_init\_private
  - Attributes (pkcs11\_attrib\_), 360
- pkcs11\_config\_init\_public
  - Attributes (pkcs11\_attrib\_), 360
- pkcs11\_config\_key
  - Attributes (pkcs11\_attrib\_), 361
  - example\_pkcs11\_config.c, 832
- pkcs11\_config\_load
  - Attributes (pkcs11\_attrib\_), 361
- pkcs11\_config\_load\_objects
  - Attributes (pkcs11\_attrib\_), 361
  - example\_pkcs11\_config.c, 832
- pkcs11\_config\_remove\_object
  - Attributes (pkcs11\_attrib\_), 361
- PKCS11\_DEBUG
  - pkcs11\_debug.h, 900
- pkcs11\_debug.c, 900
- pkcs11\_debug.h, 900
  - PKCS11\_DEBUG, 900
  - pkcs11\_debug\_attributes, 901
  - PKCS11\_DEBUG\_NOFILE, 901
  - PKCS11\_DEBUG\_RETURN, 901
- pkcs11\_debug\_attributes
  - pkcs11\_debug.h, 901
- PKCS11\_DEBUG\_NOFILE
  - pkcs11\_debug.h, 901
- PKCS11\_DEBUG\_RETURN
  - pkcs11\_debug.h, 901
- pkcs11\_deinit
  - Attributes (pkcs11\_attrib\_), 361
- pkcs11\_digest
  - pkcs11\_digest.c, 902
  - pkcs11\_digest.h, 903
- pkcs11\_digest.c, 901
  - pkcs11\_digest, 902
  - pkcs11\_digest\_final, 902
  - pkcs11\_digest\_init, 902
  - pkcs11\_digest\_update, 902
- pkcs11\_digest.h, 903
  - pkcs11\_digest, 903
  - pkcs11\_digest\_final, 903
  - pkcs11\_digest\_init, 904
  - pkcs11\_digest\_update, 904
- pkcs11\_digest\_final
  - pkcs11\_digest.c, 902
  - pkcs11\_digest.h, 903
- pkcs11\_digest\_init
  - pkcs11\_digest.c, 902
  - pkcs11\_digest.h, 904
- pkcs11\_digest\_update
  - pkcs11\_digest.c, 902
  - pkcs11\_digest.h, 904
- pkcs11\_find.c, 904
- pkcs11\_find.h, 905
- pkcs11\_find\_continue
  - Attributes (pkcs11\_attrib\_), 361
- pkcs11\_find\_finish
  - Attributes (pkcs11\_attrib\_), 362
- pkcs11\_find\_get\_attribute
  - Attributes (pkcs11\_attrib\_), 362
- pkcs11\_find\_init
  - Attributes (pkcs11\_attrib\_), 362
- pkcs11\_get\_context
  - Attributes (pkcs11\_attrib\_), 362
- pkcs11\_get\_lib\_info
  - Attributes (pkcs11\_attrib\_), 362
- pkcs11\_get\_session\_context
  - Attributes (pkcs11\_attrib\_), 362
- PKCS11\_HELPER\_DLL\_EXPORT
  - cryptoki.h, 828
- PKCS11\_HELPER\_DLL\_IMPORT
  - cryptoki.h, 828
- PKCS11\_HELPER\_DLL\_LOCAL
  - cryptoki.h, 828
- pkcs11\_info.c, 905
- pkcs11\_info.h, 906
- pkcs11\_init
  - Attributes (pkcs11\_attrib\_), 363
- pkcs11\_init.c, 907
- pkcs11\_init.h, 907
  - pkcs11\_lib\_ctx, 908
  - pkcs11\_lib\_ctx\_ptr, 908
- pkcs11\_init\_check
  - Attributes (pkcs11\_attrib\_), 363
- pkcs11\_key.c, 909
- pkcs11\_key.h, 910
- pkcs11\_key\_derive
  - Attributes (pkcs11\_attrib\_), 363
- pkcs11\_key\_ec\_private\_attributes
  - Attributes (pkcs11\_attrib\_), 374
- pkcs11\_key\_ec\_public\_attributes
  - Attributes (pkcs11\_attrib\_), 375
- pkcs11\_key\_generate\_pair
  - Attributes (pkcs11\_attrib\_), 363
- pkcs11\_key\_private\_attributes
  - Attributes (pkcs11\_attrib\_), 375
- pkcs11\_key\_private\_attributes\_count
  - Attributes (pkcs11\_attrib\_), 375
- pkcs11\_key\_public\_attributes



Attributes (pkcs11\_attrib\_), 375  
 pkcs11\_key\_public\_attributes\_count  
   Attributes (pkcs11\_attrib\_), 375  
 pkcs11\_key\_rsa\_private\_attributes  
   Attributes (pkcs11\_attrib\_), 375  
 pkcs11\_key\_secret\_attributes  
   Attributes (pkcs11\_attrib\_), 376  
 pkcs11\_key\_secret\_attributes\_count  
   Attributes (pkcs11\_attrib\_), 376  
 pkcs11\_key\_write  
   Attributes (pkcs11\_attrib\_), 364  
 pkcs11\_lib\_ctx  
   pkcs11\_init.h, 908  
 pkcs11\_lib\_ctx\_ptr  
   pkcs11\_init.h, 908  
 pkcs11\_lib\_description  
   Attributes (pkcs11\_attrib\_), 376  
 pkcs11\_lib\_manufacturer\_id  
   Attributes (pkcs11\_attrib\_), 376  
 PKCS11\_LOCAL  
   cryptoki.h, 828  
 pkcs11\_lock\_context  
   Attributes (pkcs11\_attrib\_), 364  
 pkcs11\_main.c, 910  
 pkcs11\_mech.c, 914  
 pkcs11\_mech.h, 915  
 pkcs11\_mech\_get\_list  
   Attributes (pkcs11\_attrib\_), 364  
 pkcs11\_object  
   pkcs11\_object.h, 919  
 pkcs11\_object.c, 916  
 pkcs11\_object.h, 917  
   pkcs11\_object, 919  
   pkcs11\_object\_cache\_t, 919  
   PKCS11\_OBJECT\_FLAG\_DESTROYABLE, 918  
   PKCS11\_OBJECT\_FLAG\_DYNAMIC, 918  
   PKCS11\_OBJECT\_FLAG\_MODIFIABLE, 919  
   PKCS11\_OBJECT\_FLAG\_SENSITIVE, 919  
   PKCS11\_OBJECT\_FLAG\_TA\_TYPE, 919  
   PKCS11\_OBJECT\_FLAG\_TRUST\_TYPE, 919  
   pkcs11\_object\_ptr, 919  
 pkcs11\_object\_alloc  
   Attributes (pkcs11\_attrib\_), 364  
 pkcs11\_object\_cache  
   Attributes (pkcs11\_attrib\_), 376  
 pkcs11\_object\_cache\_t  
   pkcs11\_object.h, 919  
 pkcs11\_object\_check  
   Attributes (pkcs11\_attrib\_), 364  
 pkcs11\_object\_create  
   Attributes (pkcs11\_attrib\_), 365  
 pkcs11\_object\_deinit  
   Attributes (pkcs11\_attrib\_), 365  
 pkcs11\_object\_destroy  
   Attributes (pkcs11\_attrib\_), 365  
 pkcs11\_object\_find  
   Attributes (pkcs11\_attrib\_), 365  
 PKCS11\_OBJECT\_FLAG\_DESTROYABLE  
   pkcs11\_object.h, 918  
 PKCS11\_OBJECT\_FLAG\_DYNAMIC  
   pkcs11\_object.h, 918  
 PKCS11\_OBJECT\_FLAG\_MODIFIABLE  
   pkcs11\_object.h, 919  
 PKCS11\_OBJECT\_FLAG\_SENSITIVE  
   pkcs11\_object.h, 919  
 PKCS11\_OBJECT\_FLAG\_TA\_TYPE  
   pkcs11\_object.h, 919  
 PKCS11\_OBJECT\_FLAG\_TRUST\_TYPE  
   pkcs11\_object.h, 919  
 pkcs11\_object\_free  
   Attributes (pkcs11\_attrib\_), 365  
 pkcs11\_object\_get\_class  
   Attributes (pkcs11\_attrib\_), 365  
 pkcs11\_object\_get\_destroyable  
   Attributes (pkcs11\_attrib\_), 366  
 pkcs11\_object\_get\_handle  
   Attributes (pkcs11\_attrib\_), 366  
 pkcs11\_object\_get\_name  
   Attributes (pkcs11\_attrib\_), 366  
 pkcs11\_object\_get\_size  
   Attributes (pkcs11\_attrib\_), 366  
 pkcs11\_object\_get\_type  
   Attributes (pkcs11\_attrib\_), 366  
 pkcs11\_object\_load\_handle\_info  
   Attributes (pkcs11\_attrib\_), 366  
 pkcs11\_object\_monotonic\_attributes  
   Attributes (pkcs11\_attrib\_), 376  
 pkcs11\_object\_monotonic\_attributes\_count  
   Attributes (pkcs11\_attrib\_), 377  
 pkcs11\_object\_ptr  
   pkcs11\_object.h, 919  
 pkcs11\_os.c, 920  
 pkcs11\_os.h, 920  
   pkcs11\_os\_free, 921  
   pkcs11\_os\_malloc, 921  
 pkcs11\_os\_create\_mutex  
   Attributes (pkcs11\_attrib\_), 367  
 pkcs11\_os\_destroy\_mutex  
   Attributes (pkcs11\_attrib\_), 367  
 pkcs11\_os\_free  
   pkcs11\_os.h, 921  
 pkcs11\_os\_lock\_mutex  
   Attributes (pkcs11\_attrib\_), 367  
 pkcs11\_os\_malloc  
   pkcs11\_os.h, 921  
 pkcs11\_os\_unlock\_mutex  
   Attributes (pkcs11\_attrib\_), 367  
 pkcs11\_session.c, 921  
 pkcs11\_session.h, 922  
   pkcs11\_session\_authorize, 923  
   pkcs11\_session\_ctx, 923  
   pkcs11\_session\_ctx\_ptr, 923  
 pkcs11\_session\_authorize  
   pkcs11\_session.h, 923  
 pkcs11\_session\_check  
   Attributes (pkcs11\_attrib\_), 367

- pkcs11\_session\_close
  - Attributes (pkcs11\_attrib\_), 367
- pkcs11\_session\_closeall
  - Attributes (pkcs11\_attrib\_), 368
- pkcs11\_session\_ctx
  - pkcs11\_session.h, 923
- pkcs11\_session\_ctx\_ptr
  - pkcs11\_session.h, 923
- pkcs11\_session\_get\_info
  - Attributes (pkcs11\_attrib\_), 368
- pkcs11\_session\_login
  - Attributes (pkcs11\_attrib\_), 368
- pkcs11\_session\_logout
  - Attributes (pkcs11\_attrib\_), 368
- pkcs11\_session\_open
  - Attributes (pkcs11\_attrib\_), 368
- pkcs11\_signature.c, 923
- pkcs11\_signature.h, 924
- pkcs11\_signature\_sign
  - Attributes (pkcs11\_attrib\_), 368
- pkcs11\_signature\_sign\_continue
  - Attributes (pkcs11\_attrib\_), 369
- pkcs11\_signature\_sign\_finish
  - Attributes (pkcs11\_attrib\_), 369
- pkcs11\_signature\_sign\_init
  - Attributes (pkcs11\_attrib\_), 369
- pkcs11\_signature\_verify
  - Attributes (pkcs11\_attrib\_), 369
- pkcs11\_signature\_verify\_continue
  - Attributes (pkcs11\_attrib\_), 370
- pkcs11\_signature\_verify\_finish
  - Attributes (pkcs11\_attrib\_), 370
- pkcs11\_signature\_verify\_init
  - Attributes (pkcs11\_attrib\_), 370
- pkcs11\_slot.c, 925
- pkcs11\_slot.h, 926
  - pkcs11\_slot\_ctx, 927
  - pkcs11\_slot\_ctx\_ptr, 927
- pkcs11\_slot\_config
  - Attributes (pkcs11\_attrib\_), 370
- pkcs11\_slot\_ctx
  - pkcs11\_slot.h, 927
- pkcs11\_slot\_ctx\_ptr
  - pkcs11\_slot.h, 927
- pkcs11\_slot\_get\_context
  - Attributes (pkcs11\_attrib\_), 370
- pkcs11\_slot\_get\_info
  - Attributes (pkcs11\_attrib\_), 371
- pkcs11\_slot\_get\_list
  - Attributes (pkcs11\_attrib\_), 371
- pkcs11\_slot\_init
  - Attributes (pkcs11\_attrib\_), 371
- pkcs11\_slot\_initslots
  - Attributes (pkcs11\_attrib\_), 371
- pkcs11\_token.c, 927
- pkcs11\_token.h, 928
- pkcs11\_token\_convert\_pin\_to\_key
  - Attributes (pkcs11\_attrib\_), 371
- pkcs11\_token\_get\_access\_type
  - Attributes (pkcs11\_attrib\_), 371
- pkcs11\_token\_get\_info
  - Attributes (pkcs11\_attrib\_), 372
- pkcs11\_token\_get\_storage
  - Attributes (pkcs11\_attrib\_), 372
- pkcs11\_token\_get\_writable
  - Attributes (pkcs11\_attrib\_), 372
- pkcs11\_token\_init
  - Attributes (pkcs11\_attrib\_), 372
- pkcs11\_token\_random
  - Attributes (pkcs11\_attrib\_), 372
- pkcs11\_token\_set\_pin
  - Attributes (pkcs11\_attrib\_), 372
- pkcs11\_unlock\_context
  - Attributes (pkcs11\_attrib\_), 373
- pkcs11\_util.c, 929
- pkcs11\_util.h, 929
  - PKCS11\_UTIL\_ARRAY\_SIZE, 930
- PKCS11\_UTIL\_ARRAY\_SIZE
  - pkcs11\_util.h, 930
- pkcs11\_util\_convert\_rv
  - Attributes (pkcs11\_attrib\_), 373
- pkcs11\_util\_escape\_string
  - Attributes (pkcs11\_attrib\_), 373
- pkcs11\_util\_memset
  - Attributes (pkcs11\_attrib\_), 373
- pkcs11configLABEL\_DEVICE\_CERTIFICATE\_FOR\_TLS
  - example\_pkcs11\_config.c, 831
- pkcs11configLABEL\_DEVICE\_PRIVATE\_KEY\_FOR\_TLS
  - example\_pkcs11\_config.c, 831
- pkcs11configLABEL\_DEVICE\_PUBLIC\_KEY\_FOR\_TLS
  - example\_pkcs11\_config.c, 831
- pkcs11configLABEL\_JITP\_CERTIFICATE
  - example\_pkcs11\_config.c, 832
- pkcs11f.h, 930
- pkcs11t.h, 930
  - CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS, 1038
  - CK\_AES\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR, 1038
  - CK\_AES\_CCM\_PARAMS, 1039
  - CK\_AES\_CCM\_PARAMS\_PTR, 1039
  - CK\_AES\_CTR\_PARAMS, 1039
  - CK\_AES\_CTR\_PARAMS\_PTR, 1039
  - CK\_AES\_GCM\_PARAMS, 1039
  - CK\_AES\_GCM\_PARAMS\_PTR, 1039
  - CK\_ARIA\_CBC\_ENCRYPT\_DATA\_PARAMS, 1039
  - CK\_ARIA\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR, 1039
  - CK\_ATTRIBUTE, 1040
  - CK\_ATTRIBUTE\_PTR, 1040
  - CK\_ATTRIBUTE\_TYPE, 1040
  - CK\_BBOOL, 1040
  - CK\_BYTE, 1040
  - CK\_BYTE\_PTR, 1040
  - CK\_C\_INITIALIZE\_ARGS, 1040
  - CK\_C\_INITIALIZE\_ARGS\_PTR, 1040

CK\_CALLBACK\_FUNCTION, 1062  
CK\_CAMELLIA\_CBC\_ENCRYPT\_DATA\_PARAMS, 1041  
CK\_CAMELLIA\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR, 1041  
CK\_CAMELLIA\_CTR\_PARAMS, 1041  
CK\_CAMELLIA\_CTR\_PARAMS\_PTR, 1041  
CK\_CCM\_PARAMS, 1041  
CK\_CCM\_PARAMS\_PTR, 1041  
CK\_CERTIFICATE\_CATEGORY, 1041  
CK\_CERTIFICATE\_CATEGORY\_AUTHORITY, 948  
CK\_CERTIFICATE\_CATEGORY\_OTHER\_ENTITY, 948  
CK\_CERTIFICATE\_CATEGORY\_TOKEN\_USER, 948  
CK\_CERTIFICATE\_CATEGORY\_UNSPECIFIED, 949  
CK\_CERTIFICATE\_TYPE, 1041  
CK\_CHAR, 1042  
CK\_CHAR\_PTR, 1042  
CK\_CMS\_SIG\_PARAMS, 1042  
CK\_CMS\_SIG\_PARAMS\_PTR, 1042  
CK\_DATE, 1042  
CK\_DES\_CBC\_ENCRYPT\_DATA\_PARAMS, 1042  
CK\_DES\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR, 1042  
CK\_DSA\_PARAMETER\_GEN\_PARAM, 1042  
CK\_DSA\_PARAMETER\_GEN\_PARAM\_PTR, 1043  
CK\_EC\_KDF\_TYPE, 1043  
CK\_ECDH1\_DERIVE\_PARAMS, 1043  
CK\_ECDH1\_DERIVE\_PARAMS\_PTR, 1043  
CK\_ECDH2\_DERIVE\_PARAMS, 1043  
CK\_ECDH2\_DERIVE\_PARAMS\_PTR, 1043  
CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS, 1043  
CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS\_PTR, 1043  
CK\_ECMQV\_DERIVE\_PARAMS, 1044  
CK\_ECMQV\_DERIVE\_PARAMS\_PTR, 1044  
CK\_EFFECTIVELY\_INFINITE, 949  
CK\_EXTRACT\_PARAMS, 1044  
CK\_EXTRACT\_PARAMS\_PTR, 1044  
CK\_FALSE, 949  
CK\_FLAGS, 1044  
CK\_FUNCTION\_LIST, 1044  
CK\_FUNCTION\_LIST\_PTR, 1044  
CK\_FUNCTION\_LIST\_PTR\_PTR, 1044  
CK\_GCM\_PARAMS, 1045  
CK\_GCM\_PARAMS\_PTR, 1045  
CK\_GOSTR3410\_DERIVE\_PARAMS, 1045  
CK\_GOSTR3410\_DERIVE\_PARAMS\_PTR, 1045  
CK\_GOSTR3410\_KEY\_WRAP\_PARAMS, 1045  
CK\_GOSTR3410\_KEY\_WRAP\_PARAMS\_PTR, 1045  
CK\_HW\_FEATURE\_TYPE, 1045  
CK\_INFO, 1045  
CK\_INFO\_PTR, 1046  
CK\_INVALID\_HANDLE, 949  
CK\_JAVA\_MIDP\_SECURITY\_DOMAIN, 1046  
CK\_KEA\_DERIVE\_PARAMS, 1046  
CK\_KEA\_DERIVE\_PARAMS\_PTR, 1046  
CK\_KEY\_DERIVATION\_STRING\_DATA, 1046  
CK\_KEY\_DERIVATION\_STRING\_DATA\_PTR, 1046  
CK\_KEY\_TYPE, 1046  
CK\_KEY\_WRAP\_SET\_OAEP\_PARAMS, 1046  
CK\_KEY\_WRAP\_SET\_OAEP\_PARAMS\_PTR, 1047  
CK\_KIP\_PARAMS, 1047  
CK\_KIP\_PARAMS\_PTR, 1047  
CK\_LONG, 1047  
CK\_MAC\_GENERAL\_PARAMS, 1047  
CK\_MAC\_GENERAL\_PARAMS\_PTR, 1047  
CK\_MECHANISM, 1047  
CK\_MECHANISM\_INFO, 1047  
CK\_MECHANISM\_INFO\_PTR, 1048  
CK\_MECHANISM\_PTR, 1048  
CK\_MECHANISM\_TYPE, 1048  
CK\_MECHANISM\_TYPE\_PTR, 1048  
CK\_NOTIFICATION, 1048  
CK\_OBJECT\_CLASS, 1048  
CK\_OBJECT\_CLASS\_PTR, 1048  
CK\_OBJECT\_HANDLE, 1048  
CK\_OBJECT\_HANDLE\_PTR, 1049  
CK\_OTP\_CHALLENGE, 949  
CK\_OTP\_COUNTER, 949  
CK\_OTP\_FLAGS, 949  
CK\_OTP\_FORMAT\_ALPHANUMERIC, 949  
CK\_OTP\_FORMAT\_BINARY, 950  
CK\_OTP\_FORMAT\_DECIMAL, 950  
CK\_OTP\_FORMAT\_HEXADECIMAL, 950  
CK\_OTP\_OUTPUT\_FORMAT, 950  
CK\_OTP\_OUTPUT\_LENGTH, 950  
CK\_OTP\_PARAM, 1049  
CK\_OTP\_PARAM\_IGNORED, 950  
CK\_OTP\_PARAM\_MANDATORY, 950  
CK\_OTP\_PARAM\_OPTIONAL, 950  
CK\_OTP\_PARAM\_PTR, 1049  
CK\_OTP\_PARAM\_TYPE, 1049  
CK\_OTP\_PARAMS, 1049  
CK\_OTP\_PARAMS\_PTR, 1049  
CK\_OTP\_PIN, 951  
CK\_OTP\_SIGNATURE\_INFO, 1049  
CK\_OTP\_SIGNATURE\_INFO\_PTR, 1049  
CK\_OTP\_TIME, 951  
CK\_OTP\_VALUE, 951  
CK\_PARAM\_TYPE, 1050  
CK\_PBE\_PARAMS, 1050  
CK\_PBE\_PARAMS\_PTR, 1050  
CK\_PKCS5\_PBKD2\_PARAMS, 1050  
CK\_PKCS5\_PBKD2\_PARAMS2, 1050  
CK\_PKCS5\_PBKD2\_PARAMS2\_PTR, 1050  
CK\_PKCS5\_PBKD2\_PARAMS\_PTR, 1050  
CK\_PKCS5\_PBKD2\_PSEUDO\_RANDOM\_FUNCTION\_TYPE, 1050

CK\_PKCS5\_PBKD2\_PSEUDO\_RANDOM\_FUNCTION\_TYPE, 1051  
 CK\_PKCS5\_PBKDF2\_SALT\_SOURCE\_TYPE, 1051  
 CK\_PKCS5\_PBKDF2\_SALT\_SOURCE\_TYPE\_PTR, 1051  
 CK\_RC2\_CBC\_PARAMS, 1051  
 CK\_RC2\_CBC\_PARAMS\_PTR, 1051  
 CK\_RC2\_MAC\_GENERAL\_PARAMS, 1051  
 CK\_RC2\_MAC\_GENERAL\_PARAMS\_PTR, 1051  
 CK\_RC2\_PARAMS, 1051  
 CK\_RC2\_PARAMS\_PTR, 1052  
 CK\_RC5\_CBC\_PARAMS, 1052  
 CK\_RC5\_CBC\_PARAMS\_PTR, 1052  
 CK\_RC5\_MAC\_GENERAL\_PARAMS, 1052  
 CK\_RC5\_MAC\_GENERAL\_PARAMS\_PTR, 1052  
 CK\_RC5\_PARAMS, 1052  
 CK\_RC5\_PARAMS\_PTR, 1052  
 CK\_RSA\_AES\_KEY\_WRAP\_PARAMS, 1052  
 CK\_RSA\_AES\_KEY\_WRAP\_PARAMS\_PTR, 1053  
 CK\_RSA\_PKCS\_MGF\_TYPE, 1053  
 CK\_RSA\_PKCS\_MGF\_TYPE\_PTR, 1053  
 CK\_RSA\_PKCS\_OAEP\_PARAMS, 1053  
 CK\_RSA\_PKCS\_OAEP\_PARAMS\_PTR, 1053  
 CK\_RSA\_PKCS\_OAEP\_SOURCE\_TYPE, 1053  
 CK\_RSA\_PKCS\_OAEP\_SOURCE\_TYPE\_PTR, 1053  
 CK\_RSA\_PKCS\_PSS\_PARAMS, 1053  
 CK\_RSA\_PKCS\_PSS\_PARAMS\_PTR, 1054  
 CK\_RV, 1054  
 CK\_SECURITY\_DOMAIN\_MANUFACTURER, 951  
 CK\_SECURITY\_DOMAIN\_OPERATOR, 951  
 CK\_SECURITY\_DOMAIN\_THIRD\_PARTY, 951  
 CK\_SECURITY\_DOMAIN\_UNSPECIFIED, 951  
 CK\_SEED\_CBC\_ENCRYPT\_DATA\_PARAMS, 1054  
 CK\_SEED\_CBC\_ENCRYPT\_DATA\_PARAMS\_PTR, 1054  
 CK\_SESSION\_HANDLE, 1054  
 CK\_SESSION\_HANDLE\_PTR, 1054  
 CK\_SESSION\_INFO, 1054  
 CK\_SESSION\_INFO\_PTR, 1054  
 CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS, 1055  
 CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS\_PTR, 1055  
 CK\_SKIPJACK\_RELAYX\_PARAMS, 1055  
 CK\_SKIPJACK\_RELAYX\_PARAMS\_PTR, 1055  
 CK\_SLOT\_ID, 1055  
 CK\_SLOT\_ID\_PTR, 1055  
 CK\_SLOT\_INFO, 1055  
 CK\_SLOT\_INFO\_PTR, 1055  
 CK\_SSL3\_KEY\_MAT\_OUT, 1056  
 CK\_SSL3\_KEY\_MAT\_OUT\_PTR, 1056  
 CK\_SSL3\_KEY\_MAT\_PARAMS, 1056  
 CK\_SSL3\_KEY\_MAT\_PARAMS\_PTR, 1056  
 CK\_SSL3\_MASTER\_KEY\_DERIVE\_PARAMS, 1056  
 CK\_SSL3\_MASTER\_KEY\_DERIVE\_PARAMS\_PTR, 1056  
 CK\_SSL3\_RANDOM\_DATA, 1056  
 CK\_STATE, 1056  
 CK\_TLS12\_KEY\_MAT\_PARAMS, 1057  
 CK\_TLS12\_KEY\_MAT\_PARAMS\_PTR, 1057  
 CK\_TLS12\_MASTER\_KEY\_DERIVE\_PARAMS, 1057  
 CK\_TLS12\_MASTER\_KEY\_DERIVE\_PARAMS\_PTR, 1057  
 CK\_TLS\_KDF\_PARAMS, 1057  
 CK\_TLS\_KDF\_PARAMS\_PTR, 1057  
 CK\_TLS\_MAC\_PARAMS, 1057  
 CK\_TLS\_MAC\_PARAMS\_PTR, 1057  
 CK\_TLS\_PRF\_PARAMS, 1058  
 CK\_TLS\_PRF\_PARAMS\_PTR, 1058  
 CK\_TOKEN\_INFO, 1058  
 CK\_TOKEN\_INFO\_PTR, 1058  
 CK\_TRUE, 951  
 CK\_ULONG, 1058  
 CK\_ULONG\_PTR, 1058  
 CK\_UNAVAILABLE\_INFORMATION, 952  
 CK\_USER\_TYPE, 1058  
 CK\_UTF8CHAR, 1058  
 CK\_UTF8CHAR\_PTR, 1059  
 CK\_VERSION, 1059  
 CK\_VERSION\_PTR, 1059  
 CK\_VOID\_PTR, 1059  
 CK\_VOID\_PTR\_PTR, 1059  
 CK\_WTLS\_KEY\_MAT\_OUT, 1059  
 CK\_WTLS\_KEY\_MAT\_OUT\_PTR, 1059  
 CK\_WTLS\_KEY\_MAT\_PARAMS, 1059  
 CK\_WTLS\_KEY\_MAT\_PARAMS\_PTR, 1060  
 CK\_WTLS\_MASTER\_KEY\_DERIVE\_PARAMS, 1060  
 CK\_WTLS\_MASTER\_KEY\_DERIVE\_PARAMS\_PTR, 1060  
 CK\_WTLS\_PRF\_PARAMS, 1060  
 CK\_WTLS\_PRF\_PARAMS\_PTR, 1060  
 CK\_WTLS\_RANDOM\_DATA, 1060  
 CK\_WTLS\_RANDOM\_DATA\_PTR, 1060  
 CK\_X9\_42\_DH1\_DERIVE\_PARAMS, 1060  
 CK\_X9\_42\_DH1\_DERIVE\_PARAMS\_PTR, 1061  
 CK\_X9\_42\_DH2\_DERIVE\_PARAMS, 1061  
 CK\_X9\_42\_DH2\_DERIVE\_PARAMS\_PTR, 1061  
 CK\_X9\_42\_DH\_KDF\_TYPE, 1061  
 CK\_X9\_42\_DH\_KDF\_TYPE\_PTR, 1061  
 CK\_X9\_42\_MQV\_DERIVE\_PARAMS, 1061  
 CK\_X9\_42\_MQV\_DERIVE\_PARAMS\_PTR, 1061  
 CKA\_AC\_ISSUER, 952  
 CKA\_ALLOWED\_MECHANISMS, 952  
 CKA\_ALWAYS\_AUTHENTICATE, 952  
 CKA\_ALWAYS\_SENSITIVE, 952  
 CKA\_APPLICATION, 952  
 CKA\_ATTR\_TYPES, 952  
 CKA\_AUTH\_PIN\_FLAGS, 952

CKA\_BASE, 953  
CKA\_BITS\_PER\_PIXEL, 953  
CKA\_CERTIFICATE\_CATEGORY, 953  
CKA\_CERTIFICATE\_TYPE, 953  
CKA\_CHAR\_COLUMNS, 953  
CKA\_CHAR\_ROWS, 953  
CKA\_CHAR\_SETS, 953  
CKA\_CHECK\_VALUE, 953  
CKA\_CLASS, 954  
CKA\_COEFFICIENT, 954  
CKA\_COLOR, 954  
CKA\_COPYABLE, 954  
CKA\_DECRYPT, 954  
CKA\_DEFAULT\_CMS\_ATTRIBUTES, 954  
CKA\_DERIVE, 954  
CKA\_DERIVE\_TEMPLATE, 954  
CKA\_DESTROYABLE, 955  
CKA\_EC\_PARAMS, 955  
CKA\_EC\_POINT, 955  
CKA\_ECDSA\_PARAMS, 955  
CKA\_ENCODING\_METHODS, 955  
CKA\_ENCRYPT, 955  
CKA\_END\_DATE, 955  
CKA\_EXPONENT\_1, 955  
CKA\_EXPONENT\_2, 956  
CKA\_EXTRACTABLE, 956  
CKA\_GOST28147\_PARAMS, 956  
CKA\_GOSTR3410\_PARAMS, 956  
CKA\_GOSTR3411\_PARAMS, 956  
CKA\_HAS\_RESET, 956  
CKA\_HASH\_OF\_ISSUER\_PUBLIC\_KEY, 956  
CKA\_HASH\_OF\_SUBJECT\_PUBLIC\_KEY, 956  
CKA\_HW\_FEATURE\_TYPE, 957  
CKA\_ID, 957  
CKA\_ISSUER, 957  
CKA\_JAVA\_MIDP\_SECURITY\_DOMAIN, 957  
CKA\_KEY\_GEN\_MECHANISM, 957  
CKA\_KEY\_TYPE, 957  
CKA\_LABEL, 957  
CKA\_LOCAL, 957  
CKA\_MECHANISM\_TYPE, 958  
CKA\_MIME\_TYPES, 958  
CKA\_MODIFIABLE, 958  
CKA\_MODULUS, 958  
CKA\_MODULUS\_BITS, 958  
CKA\_NAME\_HASH\_ALGORITHM, 958  
CKA\_NEVER\_EXTRACTABLE, 958  
CKA\_OBJECT\_ID, 958  
CKA\_OTP\_CHALLENGE\_REQUIREMENT, 959  
CKA\_OTP\_COUNTER, 959  
CKA\_OTP\_COUNTER\_REQUIREMENT, 959  
CKA\_OTP\_FORMAT, 959  
CKA\_OTP\_LENGTH, 959  
CKA\_OTP\_PIN\_REQUIREMENT, 959  
CKA\_OTP\_SERVICE\_IDENTIFIER, 959  
CKA\_OTP\_SERVICE\_LOGO, 959  
CKA\_OTP\_SERVICE\_LOGO\_TYPE, 960  
CKA\_OTP\_TIME, 960  
CKA\_OTP\_TIME\_INTERVAL, 960  
CKA\_OTP\_TIME\_REQUIREMENT, 960  
CKA\_OTP\_USER\_FRIENDLY\_MODE, 960  
CKA\_OTP\_USER\_IDENTIFIER, 960  
CKA\_OWNER, 960  
CKA\_PIXEL\_X, 960  
CKA\_PIXEL\_Y, 961  
CKA\_PRIME, 961  
CKA\_PRIME\_1, 961  
CKA\_PRIME\_2, 961  
CKA\_PRIME\_BITS, 961  
CKA\_PRIVATE, 961  
CKA\_PRIVATE\_EXPONENT, 961  
CKA\_PUBLIC\_EXPONENT, 961  
CKA\_PUBLIC\_KEY\_INFO, 962  
CKA\_REQUIRED\_CMS\_ATTRIBUTES, 962  
CKA\_RESET\_ON\_INIT, 962  
CKA\_RESOLUTION, 962  
CKA\_SECONDARY\_AUTH, 962  
CKA\_SENSITIVE, 962  
CKA\_SERIAL\_NUMBER, 962  
CKA\_SIGN, 962  
CKA\_SIGN\_RECOVER, 963  
CKA\_START\_DATE, 963  
CKA\_SUB\_PRIME\_BITS, 963  
CKA\_SUBJECT, 963  
CKA\_SUBPRIME, 963  
CKA\_SUBPRIME\_BITS, 963  
CKA\_SUPPORTED\_CMS\_ATTRIBUTES, 963  
CKA\_TOKEN, 963  
CKA\_TRUSTED, 964  
CKA\_UNWRAP, 964  
CKA\_UNWRAP\_TEMPLATE, 964  
CKA\_URL, 964  
CKA\_VALUE, 964  
CKA\_VALUE\_BITS, 964  
CKA\_VALUE\_LEN, 964  
CKA\_VENDOR\_DEFINED, 964  
CKA\_VERIFY, 965  
CKA\_VERIFY\_RECOVER, 965  
CKA\_WRAP, 965  
CKA\_WRAP\_TEMPLATE, 965  
CKA\_WRAP\_WITH\_TRUSTED, 965  
CKC\_OPENPGP, 965  
CKC\_VENDOR\_DEFINED, 965  
CKC\_WTLS, 965  
CKC\_X\_509, 966  
CKC\_X\_509\_ATTR\_CERT, 966  
CKD\_CPDIVERSIFY\_KDF, 966  
CKD\_NULL, 966  
CKD\_SHA1\_KDF, 966  
CKD\_SHA1\_KDF\_ASN1, 966  
CKD\_SHA1\_KDF\_CONCATENATE, 966  
CKD\_SHA224\_KDF, 966  
CKD\_SHA256\_KDF, 967  
CKD\_SHA384\_KDF, 967  
CKD\_SHA512\_KDF, 967  
CKF\_ARRAY\_ATTRIBUTE, 967

CKF\_CLOCK\_ON\_TOKEN, 967  
 CKF\_DECRYPT, 967  
 CKF\_DERIVE, 967  
 CKF\_DIGEST, 967  
 CKF\_DONT\_BLOCK, 968  
 CKF\_DUAL\_CRYPTO\_OPERATIONS, 968  
 CKF\_EC\_COMPRESS, 968  
 CKF\_EC\_ECPARAMETERS, 968  
 CKF\_EC\_F\_2M, 968  
 CKF\_EC\_F\_P, 968  
 CKF\_EC\_NAMEDCURVE, 968  
 CKF\_EC\_UNCOMPRESS, 968  
 CKF\_ENCRYPT, 969  
 CKF\_ERROR\_STATE, 969  
 CKF\_EXCLUDE\_CHALLENGE, 969  
 CKF\_EXCLUDE\_COUNTER, 969  
 CKF\_EXCLUDE\_PIN, 969  
 CKF\_EXCLUDE\_TIME, 969  
 CKF\_EXTENSION, 969  
 CKF\_GENERATE, 969  
 CKF\_GENERATE\_KEY\_PAIR, 970  
 CKF\_HW, 970  
 CKF\_HW\_SLOT, 970  
 CKF\_LIBRARY\_CANT\_CREATE\_OS\_THREADS, 970  
 CKF\_LOGIN\_REQUIRED, 970  
 CKF\_NEXT\_OTP, 970  
 CKF\_OS\_LOCKING\_OK, 970  
 CKF\_PROTECTED\_AUTHENTICATION\_PATH, 970  
 CKF\_REMOVABLE\_DEVICE, 971  
 CKF\_RESTORE\_KEY\_NOT\_NEEDED, 971  
 CKF\_RNG, 971  
 CKF\_RW\_SESSION, 971  
 CKF\_SECONDARY\_AUTHENTICATION, 971  
 CKF\_SERIAL\_SESSION, 971  
 CKF\_SIGN, 971  
 CKF\_SIGN\_RECOVER, 971  
 CKF\_SO\_PIN\_COUNT\_LOW, 972  
 CKF\_SO\_PIN\_FINAL\_TRY, 972  
 CKF\_SO\_PIN\_LOCKED, 972  
 CKF\_SO\_PIN\_TO\_BE\_CHANGED, 972  
 CKF\_TOKEN\_INITIALIZED, 972  
 CKF\_TOKEN\_PRESENT, 972  
 CKF\_UNWRAP, 972  
 CKF\_USER\_FRIENDLY\_OTP, 972  
 CKF\_USER\_PIN\_COUNT\_LOW, 973  
 CKF\_USER\_PIN\_FINAL\_TRY, 973  
 CKF\_USER\_PIN\_INITIALIZED, 973  
 CKF\_USER\_PIN\_LOCKED, 973  
 CKF\_USER\_PIN\_TO\_BE\_CHANGED, 973  
 CKF\_VERIFY, 973  
 CKF\_VERIFY\_RECOVER, 973  
 CKF\_WRAP, 973  
 CKF\_WRITE\_PROTECTED, 974  
 CKG\_MGF1\_SHA1, 974  
 CKG\_MGF1\_SHA224, 974  
 CKG\_MGF1\_SHA256, 974  
 CKH\_CLOCK, 974  
 CKH\_MONOTONIC\_COUNTER, 974  
 CKH\_USER\_INTERFACE, 975  
 CKH\_VENDOR\_DEFINED, 975  
 CKK\_ACTI, 975  
 CKK\_AES, 975  
 CKK\_ARIA, 975  
 CKK\_BATON, 975  
 CKK\_BLOWFISH, 975  
 CKK\_CAMELLIA, 975  
 CKK\_CAST, 976  
 CKK\_CAST128, 976  
 CKK\_CAST3, 976  
 CKK\_CAST5, 976  
 CKK\_CDMF, 976  
 CKK\_DES, 976  
 CKK\_DES2, 976  
 CKK\_DES3, 976  
 CKK\_DH, 977  
 CKK\_DSA, 977  
 CKK\_EC, 977  
 CKK\_ECDSA, 977  
 CKK\_GENERIC\_SECRET, 977  
 CKK\_GOST28147, 977  
 CKK\_GOSTR3410, 977  
 CKK\_GOSTR3411, 977  
 CKK\_HOTP, 978  
 CKK\_IDEA, 978  
 CKK\_JUNIPER, 978  
 CKK\_KEA, 978  
 CKK\_MD5\_HMAC, 978  
 CKK\_RC2, 978  
 CKK\_RC4, 978  
 CKK\_RC5, 978  
 CKK\_RIPEMD128\_HMAC, 979  
 CKK\_RIPEMD160\_HMAC, 979  
 CKK\_RSA, 979  
 CKK\_SECURID, 979  
 CKK\_SEED, 979  
 CKK\_SHA224\_HMAC, 979  
 CKK\_SHA256\_HMAC, 979  
 CKK\_SHA384\_HMAC, 979  
 CKK\_SHA512\_HMAC, 980  
 CKK\_SHA\_1\_HMAC, 980  
 CKK\_SKIPJACK, 980  
 CKK\_TWOFISH, 980  
 CKK\_VENDOR\_DEFINED, 980  
 CKK\_X9\_42\_DH, 980  
 CKM\_ACTI, 980  
 CKM\_ACTI\_KEY\_GEN, 980  
 CKM\_AES\_CBC, 981  
 CKM\_AES\_CBC\_ENCRYPT\_DATA, 981  
 CKM\_AES\_CBC\_PAD, 981  
 CKM\_AES\_CCM, 981  
 CKM\_AES\_CFB1, 981  
 CKM\_AES\_CFB128, 981



CKM\_AES\_CFB64, 981  
CKM\_AES\_CFB8, 981  
CKM\_AES\_CMAC, 982  
CKM\_AES\_CMAC\_GENERAL, 982  
CKM\_AES\_CTR, 982  
CKM\_AES\_CTS, 982  
CKM\_AES\_ECB, 982  
CKM\_AES\_ECB\_ENCRYPT\_DATA, 982  
CKM\_AES\_GCM, 982  
CKM\_AES\_GMAC, 982  
CKM\_AES\_KEY\_GEN, 983  
CKM\_AES\_KEY\_WRAP, 983  
CKM\_AES\_KEY\_WRAP\_PAD, 983  
CKM\_AES\_MAC, 983  
CKM\_AES\_MAC\_GENERAL, 983  
CKM\_AES\_OFB, 983  
CKM\_AES\_XCBC\_MAC, 983  
CKM\_AES\_XCBC\_MAC\_96, 983  
CKM\_ARIA\_CBC, 984  
CKM\_ARIA\_CBC\_ENCRYPT\_DATA, 984  
CKM\_ARIA\_CBC\_PAD, 984  
CKM\_ARIA\_ECB, 984  
CKM\_ARIA\_ECB\_ENCRYPT\_DATA, 984  
CKM\_ARIA\_KEY\_GEN, 984  
CKM\_ARIA\_MAC, 984  
CKM\_ARIA\_MAC\_GENERAL, 984  
CKM\_BATON\_CBC128, 985  
CKM\_BATON\_COUNTER, 985  
CKM\_BATON\_ECB128, 985  
CKM\_BATON\_ECB96, 985  
CKM\_BATON\_KEY\_GEN, 985  
CKM\_BATON\_SHUFFLE, 985  
CKM\_BATON\_WRAP, 985  
CKM\_BLOWFISH\_CBC, 985  
CKM\_BLOWFISH\_CBC\_PAD, 986  
CKM\_BLOWFISH\_KEY\_GEN, 986  
CKM\_CAMELLIA\_CBC, 986  
CKM\_CAMELLIA\_CBC\_ENCRYPT\_DATA, 986  
CKM\_CAMELLIA\_CBC\_PAD, 986  
CKM\_CAMELLIA\_CTR, 986  
CKM\_CAMELLIA\_ECB, 986  
CKM\_CAMELLIA\_ECB\_ENCRYPT\_DATA, 986  
CKM\_CAMELLIA\_KEY\_GEN, 987  
CKM\_CAMELLIA\_MAC, 987  
CKM\_CAMELLIA\_MAC\_GENERAL, 987  
CKM\_CAST128\_CBC, 987  
CKM\_CAST128\_CBC\_PAD, 987  
CKM\_CAST128\_ECB, 987  
CKM\_CAST128\_KEY\_GEN, 987  
CKM\_CAST128\_MAC, 987  
CKM\_CAST128\_MAC\_GENERAL, 988  
CKM\_CAST3\_CBC, 988  
CKM\_CAST3\_CBC\_PAD, 988  
CKM\_CAST3\_ECB, 988  
CKM\_CAST3\_KEY\_GEN, 988  
CKM\_CAST3\_MAC, 988  
CKM\_CAST3\_MAC\_GENERAL, 988  
CKM\_CAST5\_CBC, 988  
CKM\_CAST5\_CBC\_PAD, 989  
CKM\_CAST5\_ECB, 989  
CKM\_CAST5\_KEY\_GEN, 989  
CKM\_CAST5\_MAC, 989  
CKM\_CAST5\_MAC\_GENERAL, 989  
CKM\_CAST\_CBC, 989  
CKM\_CAST\_CBC\_PAD, 989  
CKM\_CAST\_ECB, 989  
CKM\_CAST\_KEY\_GEN, 990  
CKM\_CAST\_MAC, 990  
CKM\_CAST\_MAC\_GENERAL, 990  
CKM\_CDMF\_CBC, 990  
CKM\_CDMF\_CBC\_PAD, 990  
CKM\_CDMF\_ECB, 990  
CKM\_CDMF\_KEY\_GEN, 990  
CKM\_CDMF\_MAC, 990  
CKM\_CDMF\_MAC\_GENERAL, 991  
CKM\_CMS\_SIG, 991  
CKM\_CONCATENATE\_BASE\_AND\_DATA, 991  
CKM\_CONCATENATE\_BASE\_AND\_KEY, 991  
CKM\_CONCATENATE\_DATA\_AND\_BASE, 991  
CKM\_DES2\_KEY\_GEN, 991  
CKM\_DES3\_CBC, 991  
CKM\_DES3\_CBC\_ENCRYPT\_DATA, 991  
CKM\_DES3\_CBC\_PAD, 992  
CKM\_DES3\_CMAC, 992  
CKM\_DES3\_CMAC\_GENERAL, 992  
CKM\_DES3\_ECB, 992  
CKM\_DES3\_ECB\_ENCRYPT\_DATA, 992  
CKM\_DES3\_KEY\_GEN, 992  
CKM\_DES3\_MAC, 992  
CKM\_DES3\_MAC\_GENERAL, 992  
CKM\_DES\_CBC, 993  
CKM\_DES\_CBC\_ENCRYPT\_DATA, 993  
CKM\_DES\_CBC\_PAD, 993  
CKM\_DES\_CFB64, 993  
CKM\_DES\_CFB8, 993  
CKM\_DES\_ECB, 993  
CKM\_DES\_ECB\_ENCRYPT\_DATA, 993  
CKM\_DES\_KEY\_GEN, 993  
CKM\_DES\_MAC, 994  
CKM\_DES\_MAC\_GENERAL, 994  
CKM\_DES\_OFB64, 994  
CKM\_DES\_OFB8, 994  
CKM\_DH\_PKCS\_DERIVE, 994  
CKM\_DH\_PKCS\_KEY\_PAIR\_GEN, 994  
CKM\_DH\_PKCS\_PARAMETER\_GEN, 994  
CKM\_DSA, 994  
CKM\_DSA\_KEY\_PAIR\_GEN, 995  
CKM\_DSA\_PARAMETER\_GEN, 995  
CKM\_DSA\_PROBABLISTIC\_PARAMETER\_GEN, 995  
CKM\_DSA\_SHA1, 995  
CKM\_DSA\_SHA224, 995  
CKM\_DSA\_SHA256, 995  
CKM\_DSA\_SHA384, 995  
CKM\_DSA\_SHA512, 995

CKM\_DSA\_SHAWA\_TAYLOR\_PARAMETER\_GEN, 996  
 CKM\_EC\_KEY\_PAIR\_GEN, 996  
 CKM\_ECDH1\_COFACTOR\_DERIVE, 996  
 CKM\_ECDH1\_DERIVE, 996  
 CKM\_ECDH\_AES\_KEY\_WRAP, 996  
 CKM\_ECDSA, 996  
 CKM\_ECDSA\_KEY\_PAIR\_GEN, 996  
 CKM\_ECDSA\_SHA1, 996  
 CKM\_ECDSA\_SHA224, 997  
 CKM\_ECDSA\_SHA256, 997  
 CKM\_ECDSA\_SHA384, 997  
 CKM\_ECDSA\_SHA512, 997  
 CKM\_ECMQV\_DERIVE, 997  
 CKM\_EXTRACT\_KEY\_FROM\_KEY, 997  
 CKM\_FASTHASH, 997  
 CKM\_FORTEZZA\_TIMESTAMP, 997  
 CKM\_GENERIC\_SECRET\_KEY\_GEN, 998  
 CKM\_GOST28147, 998  
 CKM\_GOST28147\_ECB, 998  
 CKM\_GOST28147\_KEY\_GEN, 998  
 CKM\_GOST28147\_KEY\_WRAP, 998  
 CKM\_GOST28147\_MAC, 998  
 CKM\_GOSTR3410, 998  
 CKM\_GOSTR3410\_DERIVE, 998  
 CKM\_GOSTR3410\_KEY\_PAIR\_GEN, 999  
 CKM\_GOSTR3410\_KEY\_WRAP, 999  
 CKM\_GOSTR3410\_WITH\_GOSTR3411, 999  
 CKM\_GOSTR3411, 999  
 CKM\_GOSTR3411\_HMAC, 999  
 CKM\_HOTP, 999  
 CKM\_HOTP\_KEY\_GEN, 999  
 CKM\_IDEA\_CBC, 999  
 CKM\_IDEA\_CBC\_PAD, 1000  
 CKM\_IDEA\_ECB, 1000  
 CKM\_IDEA\_KEY\_GEN, 1000  
 CKM\_IDEA\_MAC, 1000  
 CKM\_IDEA\_MAC\_GENERAL, 1000  
 CKM\_JUNIPER\_CBC128, 1000  
 CKM\_JUNIPER\_COUNTER, 1000  
 CKM\_JUNIPER\_ECB128, 1000  
 CKM\_JUNIPER\_KEY\_GEN, 1001  
 CKM\_JUNIPER\_SHUFFLE, 1001  
 CKM\_JUNIPER\_WRAP, 1001  
 CKM\_KEA\_DERIVE, 1001  
 CKM\_KEA\_KEY\_DERIVE, 1001  
 CKM\_KEA\_KEY\_PAIR\_GEN, 1001  
 CKM\_KEY\_WRAP\_LYNKS, 1001  
 CKM\_KEY\_WRAP\_SET\_OAEP, 1001  
 CKM\_KIP\_DERIVE, 1002  
 CKM\_KIP\_MAC, 1002  
 CKM\_KIP\_WRAP, 1002  
 CKM\_MD2, 1002  
 CKM\_MD2\_HMAC, 1002  
 CKM\_MD2\_HMAC\_GENERAL, 1002  
 CKM\_MD2\_KEY\_DERIVATION, 1002  
 CKM\_MD2\_RSA\_PKCS, 1002  
 CKM\_MD5, 1003  
 CKM\_MD5\_HMAC, 1003  
 CKM\_MD5\_HMAC\_GENERAL, 1003  
 CKM\_MD5\_KEY\_DERIVATION, 1003  
 CKM\_MD5\_RSA\_PKCS, 1003  
 CKM\_PBA\_SHA1\_WITH\_SHA1\_HMAC, 1003  
 CKM\_PBE\_MD2\_DES\_CBC, 1003  
 CKM\_PBE\_MD5\_CAST128\_CBC, 1003  
 CKM\_PBE\_MD5\_CAST3\_CBC, 1004  
 CKM\_PBE\_MD5\_CAST5\_CBC, 1004  
 CKM\_PBE\_MD5\_CAST\_CBC, 1004  
 CKM\_PBE\_MD5\_DES\_CBC, 1004  
 CKM\_PBE\_SHA1\_CAST128\_CBC, 1004  
 CKM\_PBE\_SHA1\_CAST5\_CBC, 1004  
 CKM\_PBE\_SHA1\_DES2\_EDE\_CBC, 1004  
 CKM\_PBE\_SHA1\_DES3\_EDE\_CBC, 1004  
 CKM\_PBE\_SHA1\_RC2\_128\_CBC, 1005  
 CKM\_PBE\_SHA1\_RC2\_40\_CBC, 1005  
 CKM\_PBE\_SHA1\_RC4\_128, 1005  
 CKM\_PBE\_SHA1\_RC4\_40, 1005  
 CKM\_PKCS5\_PBKD2, 1005  
 CKM\_RC2\_CBC, 1005  
 CKM\_RC2\_CBC\_PAD, 1005  
 CKM\_RC2\_ECB, 1005  
 CKM\_RC2\_KEY\_GEN, 1006  
 CKM\_RC2\_MAC, 1006  
 CKM\_RC2\_MAC\_GENERAL, 1006  
 CKM\_RC4, 1006  
 CKM\_RC4\_KEY\_GEN, 1006  
 CKM\_RC5\_CBC, 1006  
 CKM\_RC5\_CBC\_PAD, 1006  
 CKM\_RC5\_ECB, 1006  
 CKM\_RC5\_KEY\_GEN, 1007  
 CKM\_RC5\_MAC, 1007  
 CKM\_RC5\_MAC\_GENERAL, 1007  
 CKM\_RIPEMD128, 1007  
 CKM\_RIPEMD128\_HMAC, 1007  
 CKM\_RIPEMD128\_HMAC\_GENERAL, 1007  
 CKM\_RIPEMD128\_RSA\_PKCS, 1007  
 CKM\_RIPEMD160, 1007  
 CKM\_RIPEMD160\_HMAC, 1008  
 CKM\_RIPEMD160\_HMAC\_GENERAL, 1008  
 CKM\_RIPEMD160\_RSA\_PKCS, 1008  
 CKM\_RSA\_9796, 1008  
 CKM\_RSA\_AES\_KEY\_WRAP, 1008  
 CKM\_RSA\_PKCS, 1008  
 CKM\_RSA\_PKCS\_KEY\_PAIR\_GEN, 1008  
 CKM\_RSA\_PKCS\_OAEP, 1008  
 CKM\_RSA\_PKCS\_OAEP\_TPM\_1\_1, 1009  
 CKM\_RSA\_PKCS\_PSS, 1009  
 CKM\_RSA\_PKCS\_TPM\_1\_1, 1009  
 CKM\_RSA\_X9\_31, 1009  
 CKM\_RSA\_X9\_31\_KEY\_PAIR\_GEN, 1009  
 CKM\_RSA\_X\_509, 1009  
 CKM\_SECURID, 1009  
 CKM\_SECURID\_KEY\_GEN, 1009  
 CKM\_SEED\_CBC, 1010  
 CKM\_SEED\_CBC\_ENCRYPT\_DATA, 1010  
 CKM\_SEED\_CBC\_PAD, 1010



CKM\_SEED\_ECB, 1010  
CKM\_SEED\_ECB\_ENCRYPT\_DATA, 1010  
CKM\_SEED\_KEY\_GEN, 1010  
CKM\_SEED\_MAC, 1010  
CKM\_SEED\_MAC\_GENERAL, 1010  
CKM\_SHA1\_KEY\_DERIVATION, 1011  
CKM\_SHA1\_RSA\_PKCS, 1011  
CKM\_SHA1\_RSA\_PKCS\_PSS, 1011  
CKM\_SHA1\_RSA\_X9\_31, 1011  
CKM\_SHA224, 1011  
CKM\_SHA224\_HMAC, 1011  
CKM\_SHA224\_HMAC\_GENERAL, 1011  
CKM\_SHA224\_KEY\_DERIVATION, 1011  
CKM\_SHA224\_RSA\_PKCS, 1012  
CKM\_SHA224\_RSA\_PKCS\_PSS, 1012  
CKM\_SHA256, 1012  
CKM\_SHA256\_HMAC, 1012  
CKM\_SHA256\_HMAC\_GENERAL, 1012  
CKM\_SHA256\_KEY\_DERIVATION, 1012  
CKM\_SHA256\_RSA\_PKCS, 1012  
CKM\_SHA256\_RSA\_PKCS\_PSS, 1012  
CKM\_SHA384, 1013  
CKM\_SHA384\_HMAC, 1013  
CKM\_SHA384\_HMAC\_GENERAL, 1013  
CKM\_SHA384\_KEY\_DERIVATION, 1013  
CKM\_SHA384\_RSA\_PKCS, 1013  
CKM\_SHA384\_RSA\_PKCS\_PSS, 1013  
CKM\_SHA512, 1013  
CKM\_SHA512\_224, 1013  
CKM\_SHA512\_224\_HMAC, 1014  
CKM\_SHA512\_224\_HMAC\_GENERAL, 1014  
CKM\_SHA512\_224\_KEY\_DERIVATION, 1014  
CKM\_SHA512\_256, 1014  
CKM\_SHA512\_256\_HMAC, 1014  
CKM\_SHA512\_256\_HMAC\_GENERAL, 1014  
CKM\_SHA512\_256\_KEY\_DERIVATION, 1014  
CKM\_SHA512\_HMAC, 1014  
CKM\_SHA512\_HMAC\_GENERAL, 1015  
CKM\_SHA512\_KEY\_DERIVATION, 1015  
CKM\_SHA512\_RSA\_PKCS, 1015  
CKM\_SHA512\_RSA\_PKCS\_PSS, 1015  
CKM\_SHA512\_T, 1015  
CKM\_SHA512\_T\_HMAC, 1015  
CKM\_SHA512\_T\_HMAC\_GENERAL, 1015  
CKM\_SHA512\_T\_KEY\_DERIVATION, 1015  
CKM\_SHA\_1, 1016  
CKM\_SHA\_1\_HMAC, 1016  
CKM\_SHA\_1\_HMAC\_GENERAL, 1016  
CKM\_SKIPJACK\_CBC64, 1016  
CKM\_SKIPJACK\_CFB16, 1016  
CKM\_SKIPJACK\_CFB32, 1016  
CKM\_SKIPJACK\_CFB64, 1016  
CKM\_SKIPJACK\_CFB8, 1016  
CKM\_SKIPJACK\_ECB64, 1017  
CKM\_SKIPJACK\_KEY\_GEN, 1017  
CKM\_SKIPJACK\_OFB64, 1017  
CKM\_SKIPJACK\_PRIVATE\_WRAP, 1017  
CKM\_SKIPJACK\_RELAYX, 1017  
CKM\_SKIPJACK\_WRAP, 1017  
CKM\_SSL3\_KEY\_AND\_MAC\_DERIVE, 1017  
CKM\_SSL3\_MASTER\_KEY\_DERIVE, 1017  
CKM\_SSL3\_MASTER\_KEY\_DERIVE\_DH, 1018  
CKM\_SSL3\_MD5\_MAC, 1018  
CKM\_SSL3\_PRE\_MASTER\_KEY\_GEN, 1018  
CKM\_SSL3\_SHA1\_MAC, 1018  
CKM\_TLS10\_MAC\_CLIENT, 1018  
CKM\_TLS10\_MAC\_SERVER, 1018  
CKM\_TLS12\_KDF, 1018  
CKM\_TLS12\_KEY\_AND\_MAC\_DERIVE, 1018  
CKM\_TLS12\_KEY\_SAFE\_DERIVE, 1019  
CKM\_TLS12\_MAC, 1019  
CKM\_TLS12\_MASTER\_KEY\_DERIVE, 1019  
CKM\_TLS12\_MASTER\_KEY\_DERIVE\_DH, 1019  
CKM\_TLS\_KDF, 1019  
CKM\_TLS\_KEY\_AND\_MAC\_DERIVE, 1019  
CKM\_TLS\_MAC, 1019  
CKM\_TLS\_MASTER\_KEY\_DERIVE, 1019  
CKM\_TLS\_MASTER\_KEY\_DERIVE\_DH, 1020  
CKM\_TLS\_PRE\_MASTER\_KEY\_GEN, 1020  
CKM\_TLS\_PRF, 1020  
CKM\_TWOFISH\_CBC, 1020  
CKM\_TWOFISH\_CBC\_PAD, 1020  
CKM\_TWOFISH\_KEY\_GEN, 1020  
CKM\_VENDOR\_DEFINED, 1020  
CKM\_WTLS\_CLIENT\_KEY\_AND\_MAC\_DERIVE, 1020  
CKM\_WTLS\_MASTER\_KEY\_DERIVE, 1021  
CKM\_WTLS\_MASTER\_KEY\_DERIVE\_DH\_ECC, 1021  
CKM\_WTLS\_PRE\_MASTER\_KEY\_GEN, 1021  
CKM\_WTLS\_PRF, 1021  
CKM\_WTLS\_SERVER\_KEY\_AND\_MAC\_DERIVE, 1021  
CKM\_X9\_42\_DH\_DERIVE, 1021  
CKM\_X9\_42\_DH\_HYBRID\_DERIVE, 1021  
CKM\_X9\_42\_DH\_KEY\_PAIR\_GEN, 1021  
CKM\_X9\_42\_DH\_PARAMETER\_GEN, 1022  
CKM\_X9\_42\_MQV\_DERIVE, 1022  
CKM\_XOR\_BASE\_AND\_DATA, 1022  
CKN\_OTP\_CHANGED, 1022  
CKN\_SURRENDER, 1022  
CKO\_CERTIFICATE, 1022  
CKO\_DATA, 1022  
CKO\_DOMAIN\_PARAMETERS, 1022  
CKO\_HW\_FEATURE, 1023  
CKO\_MECHANISM, 1023  
CKO\_OTP\_KEY, 1023  
CKO\_PRIVATE\_KEY, 1023  
CKO\_PUBLIC\_KEY, 1023  
CKO\_SECRET\_KEY, 1023  
CKO\_VENDOR\_DEFINED, 1023  
CKP\_PKCS5\_PBKD2\_HMAC\_GOSTR3411, 1023  
CKP\_PKCS5\_PBKD2\_HMAC\_SHA1, 1024  
CKP\_PKCS5\_PBKD2\_HMAC\_SHA224, 1024  
CKP\_PKCS5\_PBKD2\_HMAC\_SHA256, 1024  
CKP\_PKCS5\_PBKD2\_HMAC\_SHA384, 1024

CKP\_PKCS5\_PBKD2\_HMAC\_SHA512, [1024](#)  
 CKP\_PKCS5\_PBKD2\_HMAC\_SHA512\_224, [1024](#)  
 CKP\_PKCS5\_PBKD2\_HMAC\_SHA512\_256, [1024](#)  
 CKR\_ACTION\_PROHIBITED, [1024](#)  
 CKR\_ARGUMENTS\_BAD, [1025](#)  
 CKR\_ATTRIBUTE\_READ\_ONLY, [1025](#)  
 CKR\_ATTRIBUTE\_SENSITIVE, [1025](#)  
 CKR\_ATTRIBUTE\_TYPE\_INVALID, [1025](#)  
 CKR\_ATTRIBUTE\_VALUE\_INVALID, [1025](#)  
 CKR\_BUFFER\_TOO\_SMALL, [1025](#)  
 CKR\_CANCEL, [1025](#)  
 CKR\_CANT\_LOCK, [1025](#)  
 CKR\_CRYPTOKI\_ALREADY\_INITIALIZED, [1026](#)  
 CKR\_CRYPTOKI\_NOT\_INITIALIZED, [1026](#)  
 CKR\_CURVE\_NOT\_SUPPORTED, [1026](#)  
 CKR\_DATA\_INVALID, [1026](#)  
 CKR\_DATA\_LEN\_RANGE, [1026](#)  
 CKR\_DEVICE\_ERROR, [1026](#)  
 CKR\_DEVICE\_MEMORY, [1026](#)  
 CKR\_DEVICE\_REMOVED, [1026](#)  
 CKR\_DOMAIN\_PARAMS\_INVALID, [1027](#)  
 CKR\_ENCRYPTED\_DATA\_INVALID, [1027](#)  
 CKR\_ENCRYPTED\_DATA\_LEN\_RANGE, [1027](#)  
 CKR\_EXCEEDED\_MAX\_ITERATIONS, [1027](#)  
 CKR\_FIPS\_SELF\_TEST\_FAILED, [1027](#)  
 CKR\_FUNCTION\_CANCELED, [1027](#)  
 CKR\_FUNCTION\_FAILED, [1027](#)  
 CKR\_FUNCTION\_NOT\_PARALLEL, [1027](#)  
 CKR\_FUNCTION\_NOT\_SUPPORTED, [1028](#)  
 CKR\_FUNCTION\_REJECTED, [1028](#)  
 CKR\_GENERAL\_ERROR, [1028](#)  
 CKR\_HOST\_MEMORY, [1028](#)  
 CKR\_INFORMATION\_SENSITIVE, [1028](#)  
 CKR\_KEY\_CHANGED, [1028](#)  
 CKR\_KEY\_FUNCTION\_NOT\_PERMITTED, [1028](#)  
 CKR\_KEY\_HANDLE\_INVALID, [1028](#)  
 CKR\_KEY\_INDIGESTIBLE, [1029](#)  
 CKR\_KEY\_NEEDED, [1029](#)  
 CKR\_KEY\_NOT\_NEEDED, [1029](#)  
 CKR\_KEY\_NOT\_WRAPPABLE, [1029](#)  
 CKR\_KEY\_SIZE\_RANGE, [1029](#)  
 CKR\_KEY\_TYPE\_INCONSISTENT, [1029](#)  
 CKR\_KEY\_UNEXTRACTABLE, [1029](#)  
 CKR\_LIBRARY\_LOAD\_FAILED, [1029](#)  
 CKR\_MECHANISM\_INVALID, [1030](#)  
 CKR\_MECHANISM\_PARAM\_INVALID, [1030](#)  
 CKR\_MUTEX\_BAD, [1030](#)  
 CKR\_MUTEX\_NOT\_LOCKED, [1030](#)  
 CKR\_NEED\_TO\_CREATE\_THREADS, [1030](#)  
 CKR\_NEW\_PIN\_MODE, [1030](#)  
 CKR\_NEXT\_OTP, [1030](#)  
 CKR\_NO\_EVENT, [1030](#)  
 CKR\_OBJECT\_HANDLE\_INVALID, [1031](#)  
 CKR\_OK, [1031](#)  
 CKR\_OPERATION\_ACTIVE, [1031](#)  
 CKR\_OPERATION\_NOT\_INITIALIZED, [1031](#)  
 CKR\_PIN\_EXPIRED, [1031](#)  
 CKR\_PIN\_INCORRECT, [1031](#)  
 CKR\_PIN\_INVALID, [1031](#)  
 CKR\_PIN\_LEN\_RANGE, [1031](#)  
 CKR\_PIN\_LOCKED, [1032](#)  
 CKR\_PIN\_TOO\_WEAK, [1032](#)  
 CKR\_PUBLIC\_KEY\_INVALID, [1032](#)  
 CKR\_RANDOM\_NO\_RNG, [1032](#)  
 CKR\_RANDOM\_SEED\_NOT\_SUPPORTED, [1032](#)  
 CKR\_SAVED\_STATE\_INVALID, [1032](#)  
 CKR\_SESSION\_CLOSED, [1032](#)  
 CKR\_SESSION\_COUNT, [1032](#)  
 CKR\_SESSION\_EXISTS, [1033](#)  
 CKR\_SESSION\_HANDLE\_INVALID, [1033](#)  
 CKR\_SESSION\_PARALLEL\_NOT\_SUPPORTED, [1033](#)  
 CKR\_SESSION\_READ\_ONLY, [1033](#)  
 CKR\_SESSION\_READ\_ONLY\_EXISTS, [1033](#)  
 CKR\_SESSION\_READ\_WRITE\_SO\_EXISTS, [1033](#)  
 CKR\_SIGNATURE\_INVALID, [1033](#)  
 CKR\_SIGNATURE\_LEN\_RANGE, [1033](#)  
 CKR\_SLOT\_ID\_INVALID, [1034](#)  
 CKR\_STATE\_UNSAVEABLE, [1034](#)  
 CKR\_TEMPLATE\_INCOMPLETE, [1034](#)  
 CKR\_TEMPLATE\_INCONSISTENT, [1034](#)  
 CKR\_TOKEN\_NOT\_PRESENT, [1034](#)  
 CKR\_TOKEN\_NOT\_RECOGNIZED, [1034](#)  
 CKR\_TOKEN\_WRITE\_PROTECTED, [1034](#)  
 CKR\_UNWRAPPING\_KEY\_HANDLE\_INVALID, [1034](#)  
 CKR\_UNWRAPPING\_KEY\_SIZE\_RANGE, [1035](#)  
 CKR\_UNWRAPPING\_KEY\_TYPE\_INCONSISTENT, [1035](#)  
 CKR\_USER\_ALREADY\_LOGGED\_IN, [1035](#)  
 CKR\_USER\_ANOTHER\_ALREADY\_LOGGED\_IN, [1035](#)  
 CKR\_USER\_NOT\_LOGGED\_IN, [1035](#)  
 CKR\_USER\_PIN\_NOT\_INITIALIZED, [1035](#)  
 CKR\_USER\_TOO\_MANY\_TYPES, [1035](#)  
 CKR\_USER\_TYPE\_INVALID, [1035](#)  
 CKR\_VENDOR\_DEFINED, [1036](#)  
 CKR\_WRAPPED\_KEY\_INVALID, [1036](#)  
 CKR\_WRAPPED\_KEY\_LEN\_RANGE, [1036](#)  
 CKR\_WRAPPING\_KEY\_HANDLE\_INVALID, [1036](#)  
 CKR\_WRAPPING\_KEY\_SIZE\_RANGE, [1036](#)  
 CKR\_WRAPPING\_KEY\_TYPE\_INCONSISTENT, [1036](#)  
 CKS\_RO\_PUBLIC\_SESSION, [1036](#)  
 CKS\_RO\_USER\_FUNCTIONS, [1036](#)  
 CKS\_RW\_PUBLIC\_SESSION, [1037](#)  
 CKS\_RW\_SO\_FUNCTIONS, [1037](#)  
 CKS\_RW\_USER\_FUNCTIONS, [1037](#)  
 CKU\_CONTEXT\_SPECIFIC, [1037](#)  
 CKU\_SO, [1037](#)  
 CKU\_USER, [1037](#)  
 CKZ\_DATA\_SPECIFIED, [1037](#)  
 CKZ\_SALT\_SPECIFIED, [1037](#)  
 CRYPTOKI\_VERSION\_AMENDMENT, [1038](#)  
 CRYPTOKI\_VERSION\_MAJOR, [1038](#)

- CRYPTOKI\_VERSION\_MINOR, [1038](#)
- event, [1061](#)
- FALSE, [1038](#)
- pApplication, [1062](#)
- TRUE, [1038](#)
- pkcs\_mech\_get\_info
  - Attributes (pkcs11\_attr\_), [373](#)
- pLabel
  - CK\_TLS\_KDF\_PARAMS, [518](#)
  - CK\_TLS\_PR\_F\_PARAMS, [520](#)
  - CK\_WTLS\_PR\_F\_PARAMS, [527](#)
- pMechanism
  - CK\_KIP\_PARAMS, [492](#)
- pNewPassword
  - CK\_SKIPJACK\_RELAYX\_PARAMS, [509](#)
- pNewPublicData
  - CK\_SKIPJACK\_RELAYX\_PARAMS, [509](#)
- pNewRandomA
  - CK\_SKIPJACK\_RELAYX\_PARAMS, [509](#)
- pNonce
  - CK\_AES\_CCM\_PARAMS, [468](#)
  - CK\_CCM\_PARAMS, [475](#)
- pOAEPParams
  - CK\_RSA\_AES\_KEY\_WRAP\_PARAMS, [503](#)
- pOldPassword
  - CK\_SKIPJACK\_RELAYX\_PARAMS, [509](#)
- pOldPublicData
  - CK\_SKIPJACK\_RELAYX\_PARAMS, [510](#)
- pOldRandomA
  - CK\_SKIPJACK\_RELAYX\_PARAMS, [510](#)
- pOldWrappedX
  - CK\_SKIPJACK\_RELAYX\_PARAMS, [510](#)
- port
  - ATCAIfaceCfg, [463](#)
- pOtherInfo
  - CK\_X9\_42\_DH1\_DERIVE\_PARAMS, [529](#)
  - CK\_X9\_42\_DH2\_DERIVE\_PARAMS, [531](#)
  - CK\_X9\_42\_MQV\_DERIVE\_PARAMS, [532](#)
- pOutput
  - CK\_TLS\_PR\_F\_PARAMS, [520](#)
  - CK\_WTLS\_PR\_F\_PARAMS, [527](#)
- pParameter
  - CK\_MECHANISM, [493](#)
- pParams
  - CK\_OTP\_PARAMS, [495](#)
  - CK\_OTP\_SIGNATURE\_INFO, [495](#)
- pPassword
  - CK\_PBE\_PARAMS, [496](#)
  - CK\_PKCS5\_PBKD2\_PARAMS, [497](#)
  - CK\_PKCS5\_PBKD2\_PARAMS2, [499](#)
  - CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS, [507](#)
- pPrfData
  - CK\_PKCS5\_PBKD2\_PARAMS, [497](#)
  - CK\_PKCS5\_PBKD2\_PARAMS2, [499](#)
- pPrimeP
  - CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS, [507](#)
- pPublicData
  - CK\_ECDH1\_DERIVE\_PARAMS, [480](#)
  - CK\_ECDH2\_DERIVE\_PARAMS, [481](#)
  - CK\_ECMQV\_DERIVE\_PARAMS, [483](#)
  - CK\_GOSTR3410\_DERIVE\_PARAMS, [486](#)
  - CK\_KEA\_DERIVE\_PARAMS, [489](#)
  - CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS, [508](#)
  - CK\_X9\_42\_DH1\_DERIVE\_PARAMS, [529](#)
  - CK\_X9\_42\_DH2\_DERIVE\_PARAMS, [531](#)
  - CK\_X9\_42\_MQV\_DERIVE\_PARAMS, [532](#)
- pPublicData2
  - CK\_ECDH2\_DERIVE\_PARAMS, [481](#)
  - CK\_ECMQV\_DERIVE\_PARAMS, [483](#)
  - CK\_X9\_42\_DH2\_DERIVE\_PARAMS, [531](#)
  - CK\_X9\_42\_MQV\_DERIVE\_PARAMS, [532](#)
- pRandomA
  - CK\_KEA\_DERIVE\_PARAMS, [490](#)
  - CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS, [508](#)
- pRandomB
  - CK\_KEA\_DERIVE\_PARAMS, [490](#)
- pRequestedAttributes
  - CK\_CMS\_SIG\_PARAMS, [476](#)
- pRequiredAttributes
  - CK\_CMS\_SIG\_PARAMS, [476](#)
- pReserved
  - CK\_C\_INITIALIZE\_ARGS, [473](#)
- pReturnedKeyMaterial
  - CK\_SSL3\_KEY\_MAT\_PARAMS, [513](#)
  - CK\_TLS12\_KEY\_MAT\_PARAMS, [516](#)
  - CK\_WTLS\_KEY\_MAT\_PARAMS, [525](#)
- prf
  - CK\_PKCS5\_PBKD2\_PARAMS, [497](#)
  - CK\_PKCS5\_PBKD2\_PARAMS2, [499](#)
- prfHashMechanism
  - CK\_TLS12\_KEY\_MAT\_PARAMS, [516](#)
  - CK\_TLS12\_MASTER\_KEY\_DERIVE\_PARAMS, [517](#)
  - CK\_TLS\_MAC\_PARAMS, [519](#)
- prfMechanism
  - CK\_TLS\_KDF\_PARAMS, [518](#)
- private\_key\_slot
  - atcacert\_def\_s, [451](#)
- PRIVWRITE\_COUNT
  - calib\_command.h, [767](#)
- PRIVWRITE\_KEYID\_IDX
  - calib\_command.h, [767](#)
- PRIVWRITE\_MAC\_IDX
  - calib\_command.h, [767](#)
- PRIVWRITE\_MODE\_ENCRYPT
  - calib\_command.h, [767](#)
- PRIVWRITE\_RSP\_SIZE
  - calib\_command.h, [767](#)
- PRIVWRITE\_VALUE\_IDX
  - calib\_command.h, [768](#)
- PRIVWRITE\_ZONE\_IDX
  - calib\_command.h, [768](#)
- PRIVWRITE\_ZONE\_MASK
  - calib\_command.h, [768](#)
- pSalt
  - CK\_PBE\_PARAMS, [496](#)

- pSaltSourceData
  - CK\_PKCS5\_PBKD2\_PARAMS, [498](#)
  - CK\_PKCS5\_PBKD2\_PARAMS2, [499](#)
- pSeed
  - CK\_DSA\_PARAMETER\_GEN\_PARAM, [479](#)
  - CK\_KIP\_PARAMS, [492](#)
  - CK\_TLS\_PRf\_PARAMS, [520](#)
  - CK\_WTLS\_PRf\_PARAMS, [527](#)
- pServerRandom
  - CK\_SSL3\_RANDOM\_DATA, [515](#)
  - CK\_WTLS\_RANDOM\_DATA, [528](#)
- pSharedData
  - CK\_ECDH1\_DERIVE\_PARAMS, [480](#)
  - CK\_ECDH2\_DERIVE\_PARAMS, [481](#)
  - CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS, [482](#)
  - CK\_ECMQV\_DERIVE\_PARAMS, [484](#)
- pSigningMechanism
  - CK\_CMS\_SIG\_PARAMS, [477](#)
- pSourceData
  - CK\_RSA\_PKCS\_OAEP\_PARAMS, [504](#)
- pSubprimeQ
  - CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS, [508](#)
- public\_key
  - atca\_gen\_key\_in\_out, [425](#)
  - Host side crypto methods (atcah\_), [328](#)
- public\_key\_dev\_loc
  - atcacert\_def\_s, [451](#)
- public\_key\_size
  - atca\_gen\_key\_in\_out, [425](#)
- publicKey
  - CK\_ECMQV\_DERIVE\_PARAMS, [484](#)
  - CK\_X9\_42\_MQV\_DERIVE\_PARAMS, [533](#)
- pUKM
  - CK\_GOSTR3410\_DERIVE\_PARAMS, [487](#)
  - CK\_GOSTR3410\_KEY\_WRAP\_PARAMS, [487](#)
- pulOutputLen
  - CK\_TLS\_PRf\_PARAMS, [520](#)
  - CK\_WTLS\_PRf\_PARAMS, [528](#)
- PUNITIVE
  - license.txt, [891](#)
- pValue
  - CK\_ATTRIBUTE, [472](#)
  - CK\_OTP\_PARAM, [494](#)
- pVersion
  - CK\_SSL3\_MASTER\_KEY\_DERIVE\_PARAMS, [514](#)
  - CK\_TLS12\_MASTER\_KEY\_DERIVE\_PARAMS, [517](#)
  - CK\_WTLS\_MASTER\_KEY\_DERIVE\_PARAMS, [526](#)
- pWrapOID
  - CK\_GOSTR3410\_KEY\_WRAP\_PARAMS, [488](#)
- pX
  - CK\_KEY\_WRAP\_SET\_OAEP\_PARAMS, [491](#)
- rand\_out
  - Host side crypto methods (atcah\_), [328](#)
- RANDOM\_COUNT
  - calib\_command.h, [768](#)
- RANDOM\_MODE\_IDX
  - calib\_command.h, [768](#)
- RANDOM\_NO\_SEED\_UPDATE
  - calib\_command.h, [768](#)
- RANDOM\_NUM\_SIZE
  - calib\_command.h, [769](#)
- RANDOM\_PARAM2\_IDX
  - calib\_command.h, [769](#)
- RANDOM\_RSP\_SIZE
  - calib\_command.h, [769](#)
- RANDOM\_SEED\_UPDATE
  - calib\_command.h, [769](#)
- RandomInfo
  - CK\_SSL3\_KEY\_MAT\_PARAMS, [514](#)
  - CK\_SSL3\_MASTER\_KEY\_DERIVE\_PARAMS, [515](#)
  - CK\_TLS12\_KEY\_MAT\_PARAMS, [516](#)
  - CK\_TLS12\_MASTER\_KEY\_DERIVE\_PARAMS, [517](#)
  - CK\_TLS\_KDF\_PARAMS, [518](#)
  - CK\_WTLS\_KEY\_MAT\_PARAMS, [525](#)
  - CK\_WTLS\_MASTER\_KEY\_DERIVE\_PARAMS, [527](#)
- READ\_32\_RSP\_SIZE
  - calib\_command.h, [769](#)
- READ\_4\_RSP\_SIZE
  - calib\_command.h, [769](#)
- READ\_ADDR\_IDX
  - calib\_command.h, [770](#)
- READ\_COUNT
  - calib\_command.h, [770](#)
- read\_handle
  - hid\_device, [534](#), [535](#)
- read\_key
  - \_pkcs11\_session\_ctx, [405](#)
- READ\_ZONE\_IDX
  - calib\_command.h, [770](#)
- READ\_ZONE\_MASK
  - calib\_command.h, [770](#)
- README.md, [1063](#)
- readme.md, [1063](#)
- RECEIVE\_MODE
  - Hardware abstraction layer (hal\_), [265](#)
- ref\_ct
  - atcal2Cmaster, [458](#)
  - atcaSPImaster, [466](#)
  - atcaSWImaster, [467](#)
- releaseATCADevice
  - ATCADevice (atca\_), [145](#)
- releaseATCAIface
  - ATCAIface (atca\_), [152](#)
- reserved
  - memory\_parameters, [538](#)
- Reserved0
  - \_atecc508a\_config, [389](#)
  - \_atsha204a\_config, [396](#)
- Reserved1
  - \_atecc508a\_config, [389](#)

- \_atecc608\_config, 393
  - \_atsha204a\_config, 396
- Reserved2
  - \_atecc508a\_config, 389
  - \_atecc608\_config, 393
  - \_atsha204a\_config, 397
- Reserved3
  - \_atecc608\_config, 393
- response
  - Host side crypto methods (atcah\_), 329
- RETURN
  - calib\_aes\_gcm.c, 674
- RevNum
  - \_atecc508a\_config, 389
  - \_atecc608\_config, 393
  - \_atsha204a\_config, 397
- RFU
  - \_atecc508a\_config, 389
- rotate\_right
  - sha2\_routines.c, 1075
- RSA2048\_KEY\_SIZE
  - calib\_command.h, 770
- RX\_DELAY
  - Hardware abstraction layer (hal\_), 266
- rx\_retries
  - ATCAIfaceCfg, 463
- s\_sha\_context
  - secure\_boot\_parameters, 540
- saltSource
  - CK\_PKCS5\_PBKD2\_PARAMS, 498
  - CK\_PKCS5\_PBKD2\_PARAMS2, 499
- sam0\_change\_baudrate
  - hal\_sam0\_i2c\_asf.h, 862
- sam\_change\_baudrate
  - Hardware abstraction layer (hal\_), 269
- SCL\_PIN
  - hal\_esp32\_i2c.c, 837
- SDA\_PIN
  - hal\_esp32\_i2c.c, 837
- secure\_boot.c, 1063
  - bind\_host\_and\_secure\_element\_with\_io\_protection, 1064
  - secure\_boot\_process, 1064
- secure\_boot.h, 1065
  - bind\_host\_and\_secure\_element\_with\_io\_protection, 1066
  - host\_generate\_random\_number, 1067
  - SECURE\_BOOT\_CONFIG\_DISABLE, 1065
  - SECURE\_BOOT\_CONFIG\_FULL\_BOTH, 1066
  - SECURE\_BOOT\_CONFIG\_FULL\_DIG, 1066
  - SECURE\_BOOT\_CONFIG\_FULL\_SIGN, 1066
  - SECURE\_BOOT\_CONFIGURATION, 1066
  - SECURE\_BOOT\_DIGEST\_ENCRYPT\_ENABLED, 1066
  - secure\_boot\_process, 1067
  - SECURE\_BOOT\_UPGRADE\_SUPPORT, 1066
- secure\_boot\_check\_full\_copy\_completion
  - secure\_boot\_memory.h, 1068
- secure\_boot\_config
  - atca\_secureboot\_mac\_in\_out, 434
- secure\_boot\_config\_bits, 539
  - secure\_boot\_mode, 539
  - secure\_boot\_persistent\_enable, 539
  - secure\_boot\_pub\_key, 539
  - secure\_boot\_rand\_nonce, 539
  - secure\_boot\_reserved1, 539
  - secure\_boot\_reserved2, 539
  - secure\_boot\_sig\_dig, 540
- SECURE\_BOOT\_CONFIG\_DISABLE
  - secure\_boot.h, 1065
- SECURE\_BOOT\_CONFIG\_FULL\_BOTH
  - secure\_boot.h, 1066
- SECURE\_BOOT\_CONFIG\_FULL\_DIG
  - secure\_boot.h, 1066
- SECURE\_BOOT\_CONFIG\_FULL\_SIGN
  - secure\_boot.h, 1066
- SECURE\_BOOT\_CONFIGURATION
  - secure\_boot.h, 1066
- secure\_boot\_deinit\_memory
  - secure\_boot\_memory.h, 1068
- SECURE\_BOOT\_DIGEST\_ENCRYPT\_ENABLED
  - secure\_boot.h, 1066
- secure\_boot\_init\_memory
  - secure\_boot\_memory.h, 1068
- secure\_boot\_mark\_full\_copy\_completion
  - secure\_boot\_memory.h, 1068
- secure\_boot\_memory.h, 1067
  - secure\_boot\_check\_full\_copy\_completion, 1068
  - secure\_boot\_deinit\_memory, 1068
  - secure\_boot\_init\_memory, 1068
  - secure\_boot\_mark\_full\_copy\_completion, 1068
  - secure\_boot\_read\_memory, 1068
  - secure\_boot\_write\_memory, 1068
- secure\_boot\_mode
  - secure\_boot\_config\_bits, 539
- secure\_boot\_parameters, 540
  - app\_digest, 540
  - memory\_params, 540
  - s\_sha\_context, 540
- secure\_boot\_persistent\_enable
  - secure\_boot\_config\_bits, 539
- secure\_boot\_process
  - secure\_boot.c, 1064
  - secure\_boot.h, 1067
- secure\_boot\_pub\_key
  - secure\_boot\_config\_bits, 539
- secure\_boot\_rand\_nonce
  - secure\_boot\_config\_bits, 539
- secure\_boot\_read\_memory
  - secure\_boot\_memory.h, 1068
- secure\_boot\_reserved1
  - secure\_boot\_config\_bits, 539
- secure\_boot\_reserved2
  - secure\_boot\_config\_bits, 539
- secure\_boot\_sig\_dig
  - secure\_boot\_config\_bits, 540

- SECURE\_BOOT\_UPGRADE\_SUPPORT
  - secure\_boot.h, [1066](#)
- secure\_boot\_write\_memory
  - secure\_boot\_memory.h, [1068](#)
- SecureBoot
  - \_atecc608\_config, [393](#)
- SECUREBOOT\_COUNT\_DIG
  - calib\_command.h, [770](#)
- SECUREBOOT\_COUNT\_DIG\_SIG
  - calib\_command.h, [771](#)
- SECUREBOOT\_DIGEST\_SIZE
  - calib\_command.h, [771](#)
- SECUREBOOT\_MAC\_SIZE
  - calib\_command.h, [771](#)
- SECUREBOOT\_MODE\_ENC\_MAC\_FLAG
  - calib\_command.h, [771](#)
- SECUREBOOT\_MODE\_FULL
  - calib\_command.h, [771](#)
- SECUREBOOT\_MODE\_FULL\_COPY
  - calib\_command.h, [771](#)
- SECUREBOOT\_MODE\_FULL\_STORE
  - calib\_command.h, [772](#)
- SECUREBOOT\_MODE\_IDX
  - calib\_command.h, [772](#)
- SECUREBOOT\_MODE\_MASK
  - calib\_command.h, [772](#)
- SECUREBOOT\_MODE\_PROHIBIT\_FLAG
  - calib\_command.h, [772](#)
- SECUREBOOT\_RSP\_SIZE\_MAC
  - calib\_command.h, [772](#)
- SECUREBOOT\_RSP\_SIZE\_NO\_MAC
  - calib\_command.h, [772](#)
- SECUREBOOT\_SIGNATURE\_SIZE
  - calib\_command.h, [773](#)
- SECUREBOOTCONFIG\_MODE\_DISABLED
  - calib\_command.h, [773](#)
- SECUREBOOTCONFIG\_MODE\_FULL\_BOTH
  - calib\_command.h, [773](#)
- SECUREBOOTCONFIG\_MODE\_FULL\_DIG
  - calib\_command.h, [773](#)
- SECUREBOOTCONFIG\_MODE\_FULL\_SIG
  - calib\_command.h, [773](#)
- SECUREBOOTCONFIG\_MODE\_MASK
  - calib\_command.h, [773](#)
- SECUREBOOTCONFIG\_OFFSET
  - calib\_command.h, [774](#)
- select\_pin
  - ATCAIfaceCfg, [463](#)
- Selector
  - \_atecc508a\_config, [389](#)
  - \_atsha204a\_config, [397](#)
- SELFTEST\_COUNT
  - calib\_command.h, [774](#)
- SELFTEST\_MODE\_AES
  - calib\_command.h, [774](#)
- SELFTEST\_MODE\_ALL
  - calib\_command.h, [774](#)
- SELFTEST\_MODE\_ECDH
  - calib\_command.h, [774](#)
- SELFTEST\_MODE\_ECDSA\_SIGN\_VERIFY
  - calib\_command.h, [774](#)
- SELFTEST\_MODE\_IDX
  - calib\_command.h, [775](#)
- SELFTEST\_MODE\_RNG
  - calib\_command.h, [775](#)
- SELFTEST\_MODE\_SHA
  - calib\_command.h, [775](#)
- SELFTEST\_RSP\_SIZE
  - calib\_command.h, [775](#)
- sercom\_core\_freq
  - atcaSWImaster, [467](#)
- serialNumber
  - CK\_TOKEN\_INFO, [522](#)
- session
  - \_pkcs11\_slot\_ctx, [406](#)
- session\_counter
  - atca\_device, [420](#)
- session\_key
  - atca\_device, [420](#)
- session\_key\_id
  - atca\_device, [421](#)
- session\_key\_len
  - atca\_device, [421](#)
- session\_state
  - atca\_device, [421](#)
- sha1\_routines.c, [1069](#)
  - CL\_hash, [1069](#)
  - CL\_hashFinal, [1070](#)
  - CL\_hashInit, [1070](#)
  - CL\_hashUpdate, [1070](#)
  - shaEngine, [1071](#)
- sha1\_routines.h, [1071](#)
  - \_NOP, [1072](#)
  - \_WDRESET, [1072](#)
  - CL\_hash, [1073](#)
  - CL\_hashFinal, [1073](#)
  - CL\_hashInit, [1074](#)
  - CL\_hashUpdate, [1074](#)
  - leftRotate, [1072](#)
  - memcpy\_P, [1072](#)
  - shaEngine, [1074](#)
  - strcpy\_P, [1072](#)
  - U16, [1072](#)
  - U32, [1073](#)
  - U8, [1073](#)
- sha206a\_authenticate
  - api\_206a.c, [544](#)
  - api\_206a.h, [551](#)
- sha206a\_check\_dk\_useflag\_validity
  - api\_206a.c, [544](#)
  - api\_206a.h, [552](#)
- sha206a\_check\_pk\_useflag\_validity
  - api\_206a.c, [545](#)
  - api\_206a.h, [552](#)
- SHA206A\_DATA\_STORE0
  - api\_206a.h, [551](#)



---

SHA206A\_DATA\_STORE1  
     api\_206a.h, [551](#)  
 SHA206A\_DATA\_STORE2  
     api\_206a.h, [551](#)  
 sha206a\_diversify\_parent\_key  
     api\_206a.c, [545](#)  
     api\_206a.h, [552](#)  
 sha206a\_generate\_challenge\_response\_pair  
     api\_206a.c, [545](#)  
     api\_206a.h, [553](#)  
 sha206a\_generate\_derive\_key  
     api\_206a.c, [546](#)  
     api\_206a.h, [553](#)  
 sha206a\_get\_data\_store\_lock\_status  
     api\_206a.c, [546](#)  
     api\_206a.h, [554](#)  
 sha206a\_get\_dk\_update\_count  
     api\_206a.c, [547](#)  
     api\_206a.h, [554](#)  
 sha206a\_get\_dk\_useflag\_count  
     api\_206a.c, [547](#)  
     api\_206a.h, [554](#)  
 sha206a\_get\_pk\_useflag\_count  
     api\_206a.c, [547](#)  
     api\_206a.h, [555](#)  
 sha206a\_read\_data\_store  
     api\_206a.c, [548](#)  
     api\_206a.h, [555](#)  
 sha206a\_verify\_device\_consumption  
     api\_206a.c, [548](#)  
     api\_206a.h, [556](#)  
 sha206a\_write\_data\_store  
     api\_206a.c, [549](#)  
     api\_206a.h, [556](#)  
 SHA256\_BLOCK\_SIZE  
     sha2\_routines.h, [1077](#)  
 SHA256\_DIGEST\_SIZE  
     sha2\_routines.h, [1078](#)  
 sha2\_routines.c, [1074](#)  
     rotate\_right, [1075](#)  
     sw\_sha256, [1075](#)  
     sw\_sha256\_final, [1076](#)  
     sw\_sha256\_init, [1076](#)  
     sw\_sha256\_update, [1076](#)  
 sha2\_routines.h, [1077](#)  
     SHA256\_BLOCK\_SIZE, [1077](#)  
     SHA256\_DIGEST\_SIZE, [1078](#)  
     sw\_sha256, [1078](#)  
     sw\_sha256\_final, [1078](#)  
     sw\_sha256\_init, [1078](#)  
     sw\_sha256\_update, [1079](#)  
 SHA\_CONTEXT\_MAX\_SIZE  
     Basic Crypto API methods (atcab\_), [59](#)  
 SHA\_COUNT\_LONG  
     calib\_command.h, [775](#)  
 SHA\_COUNT\_SHORT  
     calib\_command.h, [775](#)  
 SHA\_DATA\_MAX  
     calib\_command.h, [776](#)  
 SHA\_MODE\_608\_HMAC\_END  
     calib\_command.h, [776](#)  
 SHA\_MODE\_HMAC\_END  
     calib\_command.h, [776](#)  
 SHA\_MODE\_HMAC\_START  
     calib\_command.h, [776](#)  
 SHA\_MODE\_HMAC\_UPDATE  
     calib\_command.h, [776](#)  
 SHA\_MODE\_MASK  
     calib\_command.h, [776](#)  
 SHA\_MODE\_READ\_CONTEXT  
     calib\_command.h, [777](#)  
 SHA\_MODE\_SHA256\_END  
     calib\_command.h, [777](#)  
 SHA\_MODE\_SHA256\_PUBLIC  
     calib\_command.h, [777](#)  
 SHA\_MODE\_SHA256\_START  
     calib\_command.h, [777](#)  
 SHA\_MODE\_SHA256\_UPDATE  
     calib\_command.h, [777](#)  
 SHA\_MODE\_TARGET\_MASK  
     calib\_command.h, [777](#)  
 SHA\_MODE\_TARGET\_MSGDIGBUF  
     cryptoauthlib.h, [826](#)  
 SHA\_MODE\_TARGET\_OUT\_ONLY  
     cryptoauthlib.h, [826](#)  
 SHA\_MODE\_TARGET\_TEMPKEY  
     cryptoauthlib.h, [826](#)  
 SHA\_MODE\_WRITE\_CONTEXT  
     calib\_command.h, [778](#)  
 SHA\_RSP\_SIZE  
     calib\_command.h, [778](#)  
 SHA\_RSP\_SIZE\_LONG  
     calib\_command.h, [778](#)  
 SHA\_RSP\_SIZE\_SHORT  
     calib\_command.h, [778](#)  
 shaEngine  
     sha1\_routines.c, [1071](#)  
     sha1\_routines.h, [1074](#)  
 SIGN\_COUNT  
     calib\_command.h, [778](#)  
 SIGN\_KEYID\_IDX  
     calib\_command.h, [778](#)  
 SIGN\_MODE\_EXTERNAL  
     calib\_command.h, [779](#)  
 SIGN\_MODE\_IDX  
     calib\_command.h, [779](#)  
 SIGN\_MODE\_INCLUDE\_SN  
     calib\_command.h, [779](#)  
 SIGN\_MODE\_INTERNAL  
     calib\_command.h, [779](#)  
 SIGN\_MODE\_INVALIDATE  
     calib\_command.h, [779](#)  
 SIGN\_MODE\_MASK  
     calib\_command.h, [779](#)  
 SIGN\_MODE\_SOURCE\_MASK  
     calib\_command.h, [780](#)

- SIGN\_MODE\_SOURCE\_MSGDIGBUF
  - calib\_command.h, [780](#)
- SIGN\_MODE\_SOURCE\_TEMPKEY
  - calib\_command.h, [780](#)
- SIGN\_RSP\_SIZE
  - calib\_command.h, [780](#)
- signature
  - atca\_secureboot\_mac\_in\_out, [434](#)
  - atca\_verify\_mac, [443](#)
  - Host side crypto methods (atcah\_), [329](#)
  - memory\_parameters, [538](#)
- size
  - \_pkcs11\_object, [402](#)
- slave\_address
  - ATCAIfaceCfg, [463](#)
- sLen
  - CK\_RSA\_PKCS\_PSS\_PARAMS, [505](#)
- slot
  - \_pkcs11\_object, [402](#)
  - \_pkcs11\_session\_ctx, [405](#)
  - atcacert\_device\_loc\_s, [453](#)
- slot\_cnt
  - \_pkcs11\_lib\_ctx, [400](#)
- slot\_conf
  - atca\_gen\_dig\_in\_out, [422](#)
- slot\_config
  - atca\_sign\_internal\_in\_out, [437](#)
- slot\_id
  - \_pkcs11\_slot\_ctx, [406](#)
- slot\_key
  - atca\_check\_mac\_in\_out, [415](#)
- slot\_locked
  - atca\_gen\_dig\_in\_out, [423](#)
- SlotConfig
  - \_atecc508a\_config, [389](#)
  - \_atecc608\_config, [393](#)
  - \_atsha204a\_config, [397](#)
- slotDescription
  - CK\_SLOT\_INFO, [512](#)
- slotID
  - CK\_SESSION\_INFO, [506](#)
- SlotLocked
  - \_atecc508a\_config, [390](#)
  - \_atecc608\_config, [394](#)
- slots
  - \_pkcs11\_lib\_ctx, [400](#)
- sn
  - atca\_check\_mac\_in\_out, [415](#)
  - atca\_derive\_key\_in\_out, [417](#)
  - atca\_derive\_key\_mac\_in\_out, [419](#)
  - atca\_gen\_dig\_in\_out, [423](#)
  - atca\_gen\_key\_in\_out, [425](#)
  - atca\_sign\_internal\_in\_out, [438](#)
  - atca\_verify\_mac, [443](#)
  - atca\_write\_mac\_in\_out, [444](#)
  - Host side crypto methods (atcah\_), [329](#)
- SN03
  - \_atecc508a\_config, [390](#)
- \_atecc608\_config, [394](#)
- \_atsha204a\_config, [397](#)
- SN47
  - \_atecc508a\_config, [390](#)
  - \_atecc608\_config, [394](#)
  - \_atsha204a\_config, [397](#)
- SN8
  - \_atecc508a\_config, [390](#)
  - \_atecc608\_config, [394](#)
  - \_atsha204a\_config, [397](#)
- sn\_source
  - atcacert\_def\_s, [452](#)
- SNSRC\_DEVICE\_SN
  - Certificate manipulation methods (atcacert\_), [165](#)
- SNSRC\_DEVICE\_SN\_HASH
  - Certificate manipulation methods (atcacert\_), [165](#)
- SNSRC\_DEVICE\_SN\_HASH\_POS
  - Certificate manipulation methods (atcacert\_), [165](#)
- SNSRC\_DEVICE\_SN\_HASH\_RAW
  - Certificate manipulation methods (atcacert\_), [165](#)
- SNSRC\_PUB\_KEY\_HASH
  - Certificate manipulation methods (atcacert\_), [165](#)
- SNSRC\_PUB\_KEY\_HASH\_POS
  - Certificate manipulation methods (atcacert\_), [165](#)
- SNSRC\_PUB\_KEY\_HASH\_RAW
  - Certificate manipulation methods (atcacert\_), [165](#)
- SNSRC\_SIGNER\_ID
  - Certificate manipulation methods (atcacert\_), [165](#)
- SNSRC\_STORED
  - Certificate manipulation methods (atcacert\_), [165](#)
- SNSRC\_STORED\_DYNAMIC
  - Certificate manipulation methods (atcacert\_), [165](#)
- so\_pin\_handle
  - \_pkcs11\_slot\_ctx, [407](#)
- SOFTWARE
  - license.txt, [892](#)
- software
  - license.txt, [883](#)
- Software crypto methods (atcac\_), [251](#)
  - ATCA\_ECC\_P256\_FIELD\_SIZE, [251](#)
  - ATCA\_ECC\_P256\_PRIVATE\_KEY\_SIZE, [252](#)
  - ATCA\_ECC\_P256\_PUBLIC\_KEY\_SIZE, [252](#)
  - ATCA\_ECC\_P256\_SIGNATURE\_SIZE, [252](#)
  - atcac\_sha256\_hmac\_finish, [252](#)
  - atcac\_sha256\_hmac\_init, [252](#)
  - atcac\_sha256\_hmac\_update, [253](#)
  - atcac\_sw\_ecdsa\_verify\_p256, [253](#)
  - atcac\_sw\_random, [253](#)
  - atcac\_sw\_sha1, [254](#)
  - atcac\_sw\_sha1\_finish, [254](#)
  - atcac\_sw\_sha1\_init, [254](#)
  - atcac\_sw\_sha1\_update, [255](#)
  - atcac\_sw\_sha2\_256, [255](#)
  - atcac\_sw\_sha2\_256\_finish, [255](#)
  - atcac\_sw\_sha2\_256\_init, [255](#)
  - atcac\_sw\_sha2\_256\_update, [256](#)
- source
  - CK\_RSA\_PKCS\_OAEP\_PARAMS, [504](#)



source\_flag  
     atca\_temp\_key, 440  
 SPECIAL  
     license.txt, 892  
 spi\_file  
     atcaSPImaster, 466  
 start\_address  
     memory\_parameters, 538  
 start\_change\_baudrate  
     Hardware abstraction layer (hal\_), 269  
 state  
     \_pkcs11\_session\_ctx, 405  
     CK\_SESSION\_INFO, 506  
 STATUTORY  
     license.txt, 892  
 std\_cert\_elements  
     atcacert\_def\_s, 452  
 STDCERT\_AUTH\_KEY\_ID  
     Certificate manipulation methods (atcacert\_), 166  
 STDCERT\_CERT\_SN  
     Certificate manipulation methods (atcacert\_), 166  
 STDCERT\_EXPIRE\_DATE  
     Certificate manipulation methods (atcacert\_), 166  
 STDCERT\_ISSUE\_DATE  
     Certificate manipulation methods (atcacert\_), 166  
 STDCERT\_NUM\_ELEMENTS  
     Certificate manipulation methods (atcacert\_), 166  
 STDCERT\_PUBLIC\_KEY  
     Certificate manipulation methods (atcacert\_), 166  
 STDCERT\_SIGNATURE  
     Certificate manipulation methods (atcacert\_), 166  
 STDCERT\_SIGNER\_ID  
     Certificate manipulation methods (atcacert\_), 166  
 STDCERT\_SUBJ\_KEY\_ID  
     Certificate manipulation methods (atcacert\_), 166  
 stopbits  
     ATCAIfaceCfg, 464  
 stored\_value  
     atca\_gen\_dig\_in\_out, 423  
 strcpy\_P  
     sha1\_routines.h, 1072  
 strnchr  
     Hardware abstraction layer (hal\_), 304  
 sw\_sha256  
     sha2\_routines.c, 1075  
     sha2\_routines.h, 1078  
 sw\_sha256\_ctx, 540  
     block, 541  
     block\_size, 541  
     hash, 541  
     total\_msg\_size, 541  
 sw\_sha256\_final  
     sha2\_routines.c, 1076  
     sha2\_routines.h, 1078  
 sw\_sha256\_init  
     sha2\_routines.c, 1076  
     sha2\_routines.h, 1078  
 sw\_sha256\_update  
     sha2\_routines.c, 1076  
     sha2\_routines.h, 1079  
 SWI\_FLAG\_CMD  
     Hardware abstraction layer (hal\_), 266  
 SWI\_FLAG\_IDLE  
     Hardware abstraction layer (hal\_), 266  
 SWI\_FLAG\_SLEEP  
     Hardware abstraction layer (hal\_), 266  
 SWI\_FLAG\_TX  
     Hardware abstraction layer (hal\_), 266  
 swi\_uart\_deinit  
     Hardware abstraction layer (hal\_), 304  
 swi\_uart\_discover\_buses  
     Hardware abstraction layer (hal\_), 304  
 swi\_uart\_init  
     Hardware abstraction layer (hal\_), 305  
 swi\_uart\_mode  
     Hardware abstraction layer (hal\_), 305  
 swi\_uart\_receive\_byte  
     Hardware abstraction layer (hal\_), 306  
 swi\_uart\_samd21\_asf.c, 1079  
 swi\_uart\_samd21\_asf.h, 1080  
 swi\_uart\_send\_byte  
     Hardware abstraction layer (hal\_), 306  
 swi\_uart\_setbaud  
     Hardware abstraction layer (hal\_), 306  
 swi\_uart\_start.c, 1081  
     USART\_BAUD\_RATE, 1082  
 swi\_uart\_start.h, 1082  
 SWI\_WAKE\_TOKEN  
     Hardware abstraction layer (hal\_), 266  
 symmetric\_authenticate  
     symmetric\_authentication.c, 1084  
     symmetric\_authentication.h, 1085  
 symmetric\_authentication.c, 1083  
     symmetric\_authenticate, 1084  
 symmetric\_authentication.h, 1084  
     symmetric\_authenticate, 1085  
 systemd  
     license.txt, 893  
 TA100  
     ATCADevice (atca\_), 143  
 TABLE\_SIZE  
     Attributes (pkcs11\_attrib\_), 342  
 TAG  
     hal\_esp32\_i2c.c, 847  
 target\_key  
     atca\_check\_mac\_in\_out, 415  
     atca\_derive\_key\_in\_out, 418  
 target\_key\_id  
     atca\_derive\_key\_in\_out, 418  
     atca\_derive\_key\_mac\_in\_out, 419  
 tbs\_cert\_loc  
     atcacert\_def\_s, 452  
 temp\_key  
     atca\_check\_mac\_in\_out, 415  
     atca\_derive\_key\_in\_out, 418  
     atca\_gen\_dig\_in\_out, 423

- atca\_gen\_key\_in\_out, [425](#)
- atca\_secureboot\_enc\_in\_out, [433](#)
- atca\_sign\_internal\_in\_out, [438](#)
- atca\_verify\_mac, [443](#)
- atca\_write\_mac\_in\_out, [444](#)
- Host side crypto methods (atcah\_), [330](#)
- template\_id
  - atcacert\_def\_s, [452](#)
- terms
  - license.txt, [893](#)
- TF\_BIN2HEX\_LC
  - Certificate manipulation methods (atcacert\_), [168](#)
- TF\_BIN2HEX\_SPACE\_LC
  - Certificate manipulation methods (atcacert\_), [168](#)
- TF\_BIN2HEX\_SPACE\_UC
  - Certificate manipulation methods (atcacert\_), [168](#)
- TF\_BIN2HEX\_UC
  - Certificate manipulation methods (atcacert\_), [168](#)
- TF\_HEX2BIN\_LC
  - Certificate manipulation methods (atcacert\_), [168](#)
- TF\_HEX2BIN\_SPACE\_LC
  - Certificate manipulation methods (atcacert\_), [168](#)
- TF\_HEX2BIN\_SPACE\_UC
  - Certificate manipulation methods (atcacert\_), [168](#)
- TF\_HEX2BIN\_UC
  - Certificate manipulation methods (atcacert\_), [168](#)
- TF\_NONE
  - Certificate manipulation methods (atcacert\_), [168](#)
- TF\_REVERSE
  - Certificate manipulation methods (atcacert\_), [168](#)
- tflxtls\_cert\_def\_4\_device.c, [1085](#)
  - g\_tflxtls\_cert\_elements\_4\_device, [1086](#)
  - g\_tflxtls\_cert\_template\_4\_device, [1086](#)
- tflxtls\_cert\_def\_4\_device.h, [1086](#)
- tm\_hour
  - atcacert\_tm\_utc\_s, [454](#)
- tm\_mday
  - atcacert\_tm\_utc\_s, [454](#)
- tm\_min
  - atcacert\_tm\_utc\_s, [454](#)
- tm\_mon
  - atcacert\_tm\_utc\_s, [455](#)
- tm\_sec
  - atcacert\_tm\_utc\_s, [455](#)
- tm\_year
  - atcacert\_tm\_utc\_s, [455](#)
- TNG API (tng\_), [378](#)
  - CRYPTOAUTH\_ROOT\_CA\_002\_PUBLIC\_KEY\_OFFSET, [379](#)
  - g\_cryptoauth\_root\_ca\_002\_cert, [384](#)
  - g\_cryptoauth\_root\_ca\_002\_cert\_size, [384](#)
  - g\_tnglora\_cert\_def\_1\_signer, [384](#)
  - g\_tnglora\_cert\_def\_2\_device, [384](#)
  - g\_tnglora\_cert\_def\_4\_device, [385](#)
  - g\_tngtls\_cert\_def\_1\_signer, [385](#)
  - g\_tngtls\_cert\_def\_2\_device, [385](#)
  - g\_tngtls\_cert\_def\_3\_device, [385](#)
  - tng\_atcacert\_device\_public\_key, [379](#)
  - tng\_atcacert\_max\_device\_cert\_size, [380](#)
  - tng\_atcacert\_max\_signer\_cert\_size, [380](#)
  - tng\_atcacert\_read\_device\_cert, [380](#)
  - tng\_atcacert\_read\_signer\_cert, [381](#)
  - tng\_atcacert\_root\_cert, [381](#)
  - tng\_atcacert\_root\_cert\_size, [382](#)
  - tng\_atcacert\_root\_public\_key, [382](#)
  - tng\_atcacert\_signer\_public\_key, [382](#)
  - tng\_get\_device\_cert\_def, [383](#)
  - tng\_get\_device\_pubkey, [383](#)
  - tng\_map\_get\_device\_cert\_def, [384](#)
  - TNGLORA\_CERT\_TEMPLATE\_4\_DEVICE\_SIZE, [379](#)
  - TNGTLS\_CERT\_ELEMENTS\_2\_DEVICE\_COUNT, [379](#)
  - TNGTLS\_CERT\_TEMPLATE\_1\_SIGNER\_SIZE, [379](#)
  - TNGTLS\_CERT\_TEMPLATE\_2\_DEVICE\_SIZE, [379](#)
  - TNGTLS\_CERT\_TEMPLATE\_3\_DEVICE\_SIZE, [379](#)
- tng\_atca.c, [1087](#)
- tng\_atca.h, [1087](#)
- tng\_atcacert\_client.c, [1088](#)
  - tng\_atcacert\_device\_public\_key, [1089](#)
  - tng\_atcacert\_max\_signer\_cert\_size, [1089](#)
  - tng\_atcacert\_read\_device\_cert, [1090](#)
  - tng\_atcacert\_read\_signer\_cert, [1090](#)
  - tng\_atcacert\_root\_cert, [1090](#)
  - tng\_atcacert\_root\_cert\_size, [1091](#)
  - tng\_atcacert\_root\_public\_key, [1091](#)
  - tng\_atcacert\_signer\_public\_key, [1092](#)
- tng\_atcacert\_client.h, [1092](#)
- tng\_atcacert\_device\_public\_key
  - TNG API (tng\_), [379](#)
  - tng\_atcacert\_client.c, [1089](#)
- tng\_atcacert\_max\_device\_cert\_size
  - TNG API (tng\_), [380](#)
- tng\_atcacert\_max\_signer\_cert\_size
  - TNG API (tng\_), [380](#)
  - tng\_atcacert\_client.c, [1089](#)
- tng\_atcacert\_read\_device\_cert
  - TNG API (tng\_), [380](#)
  - tng\_atcacert\_client.c, [1090](#)
- tng\_atcacert\_read\_signer\_cert
  - TNG API (tng\_), [381](#)
- tng\_atcacert\_root\_cert
  - TNG API (tng\_), [381](#)
  - tng\_atcacert\_client.c, [1090](#)
- tng\_atcacert\_root\_cert\_size
  - TNG API (tng\_), [382](#)
  - tng\_atcacert\_client.c, [1091](#)
- tng\_atcacert\_root\_public\_key
  - TNG API (tng\_), [382](#)
  - tng\_atcacert\_client.c, [1091](#)
- tng\_atcacert\_signer\_public\_key

- TNG API (tng\_), 382
- tng\_atcacert\_client.c, 1092
- tng\_cert\_map\_element, 542
  - cert\_def, 542
  - opcode, 542
- tng\_get\_device\_cert\_def
  - TNG API (tng\_), 383
- tng\_get\_device\_pubkey
  - TNG API (tng\_), 383
- tng\_map\_get\_device\_cert\_def
  - TNG API (tng\_), 384
- tng\_root\_cert.c, 1093
  - g\_cryptoauth\_root\_ca\_002\_cert, 1093
  - g\_cryptoauth\_root\_ca\_002\_cert\_size, 1094
- tng\_root\_cert.h, 1094
- tnglora\_cert\_def\_1\_signer.c, 1094
  - g\_tngtls\_cert\_elements\_1\_signer, 1095
  - g\_tngtls\_cert\_template\_1\_signer, 1095
- tnglora\_cert\_def\_1\_signer.h, 1095
- tnglora\_cert\_def\_2\_device.c, 1096
  - g\_tngtls\_cert\_elements\_2\_device, 1096
  - g\_tngtls\_cert\_template\_2\_device, 1096
- tnglora\_cert\_def\_2\_device.h, 1096
- tnglora\_cert\_def\_4\_device.c, 1097
  - g\_tnglora\_cert\_def\_4\_device, 1097
  - g\_tnglora\_cert\_elements\_4\_device, 1097
  - g\_tnglora\_cert\_template\_4\_device, 1098
- tnglora\_cert\_def\_4\_device.h, 1098
- TNGLORA\_CERT\_TEMPLATE\_4\_DEVICE\_SIZE
  - TNG API (tng\_), 379
- tngtls\_cert\_def\_1\_signer.c, 1098
  - g\_tngtls\_cert\_def\_1\_signer, 1099
  - g\_tngtls\_cert\_elements\_1\_signer, 1099
  - g\_tngtls\_cert\_template\_1\_signer, 1099
- tngtls\_cert\_def\_1\_signer.h, 1099
- tngtls\_cert\_def\_2\_device.c, 1100
  - g\_tngtls\_cert\_def\_2\_device, 1100
  - g\_tngtls\_cert\_elements\_2\_device, 1100
  - g\_tngtls\_cert\_template\_2\_device, 1100
- tngtls\_cert\_def\_2\_device.h, 1101
- tngtls\_cert\_def\_3\_device.c, 1101
  - g\_tngtls\_cert\_def\_3\_device, 1102
  - g\_tngtls\_cert\_elements\_3\_device, 1102
  - g\_tngtls\_cert\_template\_3\_device, 1102
- tngtls\_cert\_def\_3\_device.h, 1102
- TNGTLS\_CERT\_ELEMENTS\_2\_DEVICE\_COUNT
  - TNG API (tng\_), 379
- TNGTLS\_CERT\_TEMPLATE\_1\_SIGNER\_SIZE
  - TNG API (tng\_), 379
- TNGTLS\_CERT\_TEMPLATE\_2\_DEVICE\_SIZE
  - TNG API (tng\_), 379
- TNGTLS\_CERT\_TEMPLATE\_3\_DEVICE\_SIZE
  - TNG API (tng\_), 379
- TO
  - license.txt, 893
- TORT
  - license.txt, 883
- total\_msg\_size
  - atca\_sha256\_ctx, 435
  - hw\_sha256\_ctx, 536
  - sw\_sha256\_ctx, 541
- transforms
  - atcacert\_cert\_element\_s, 447
- TRANSMIT\_MODE
  - Hardware abstraction layer (hal\_), 267
- TRUE
  - Certificate manipulation methods (atcacert\_), 162
  - pkcs11t.h, 1038
- trust\_pkcs11\_config.c, 1102
- twi\_id
  - atcal2Cmaster, 458
- twi\_master\_instance
  - atcal2Cmaster, 459
- TX\_DELAY
  - Hardware abstraction layer (hal\_), 267
- txsize
  - ATCAPacket, 465
- type
  - \_pkcs11\_mech\_table\_e, 398
  - \_pkcs11\_attr\_model, 398
  - atcacert\_def\_s, 452
  - CK\_ATTRIBUTE, 472
  - CK\_OTP\_PARAM, 494
- U16
  - sha1\_routines.h, 1072
- U32
  - sha1\_routines.h, 1073
- U8
  - sha1\_routines.h, 1073
- ulAADLen
  - CK\_AES\_CCM\_PARAMS, 468
  - CK\_AES\_GCM\_PARAMS, 470
  - CK\_CCM\_PARAMS, 475
  - CK\_GCM\_PARAMS, 485
- ulAESKeyBits
  - CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS, 482
  - CK\_RSA\_AES\_KEY\_WRAP\_PARAMS, 503
- ulClientRandomLen
  - CK\_SSL3\_RANDOM\_DATA, 515
  - CK\_WTLS\_RANDOM\_DATA, 529
- ulContextDataLength
  - CK\_TLS\_KDF\_PARAMS, 518
- ulCount
  - CK\_OTP\_PARAMS, 495
  - CK\_OTP\_SIGNATURE\_INFO, 495
- ulCounterBits
  - CK\_AES\_CTR\_PARAMS, 469
  - CK\_CAMELLIA\_CTR\_PARAMS, 474
- ulDataLen
  - CK\_AES\_CCM\_PARAMS, 469
  - CK\_CCM\_PARAMS, 475
- ulDeviceError
  - CK\_SESSION\_INFO, 507
- ulEffectiveBits
  - CK\_RC2\_CBC\_PARAMS, 500
  - CK\_RC2\_MAC\_GENERAL\_PARAMS, 501

- ulFreePrivateMemory
  - CK\_TOKEN\_INFO, [522](#)
- ulFreePublicMemory
  - CK\_TOKEN\_INFO, [522](#)
- ulIndex
  - CK\_DSA\_PARAMETER\_GEN\_PARAM, [479](#)
- ulIteration
  - CK\_PBE\_PARAMS, [496](#)
- ulIvBits
  - CK\_AES\_GCM\_PARAMS, [470](#)
  - CK\_GCM\_PARAMS, [486](#)
- ulIvLen
  - CK\_AES\_GCM\_PARAMS, [470](#)
  - CK\_GCM\_PARAMS, [486](#)
  - CK\_RC5\_CBC\_PARAMS, [501](#)
- ulIVSizeInBits
  - CK\_SSL3\_KEY\_MAT\_PARAMS, [514](#)
  - CK\_TLS12\_KEY\_MAT\_PARAMS, [516](#)
  - CK\_WTLS\_KEY\_MAT\_PARAMS, [525](#)
- ulKeySizeInBits
  - CK\_SSL3\_KEY\_MAT\_PARAMS, [514](#)
  - CK\_TLS12\_KEY\_MAT\_PARAMS, [517](#)
  - CK\_WTLS\_KEY\_MAT\_PARAMS, [526](#)
- ulLabelLen
  - CK\_TLS\_PRF\_PARAMS, [520](#)
  - CK\_WTLS\_PRF\_PARAMS, [528](#)
- ulLabelLength
  - CK\_TLS\_KDF\_PARAMS, [519](#)
- ulLen
  - CK\_KEY\_DERIVATION\_STRING\_DATA, [491](#)
- ulMACLen
  - CK\_AES\_CCM\_PARAMS, [469](#)
  - CK\_CCM\_PARAMS, [475](#)
- ulMacLength
  - CK\_RC2\_MAC\_GENERAL\_PARAMS, [501](#)
  - CK\_RC5\_MAC\_GENERAL\_PARAMS, [502](#)
  - CK\_TLS\_MAC\_PARAMS, [519](#)
- ulMacSizeInBits
  - CK\_SSL3\_KEY\_MAT\_PARAMS, [514](#)
  - CK\_TLS12\_KEY\_MAT\_PARAMS, [517](#)
  - CK\_WTLS\_KEY\_MAT\_PARAMS, [526](#)
- ulMaxKeySize
  - CK\_MECHANISM\_INFO, [493](#)
- ulMaxPinLen
  - CK\_TOKEN\_INFO, [522](#)
- ulMaxRwSessionCount
  - CK\_TOKEN\_INFO, [522](#)
- ulMaxSessionCount
  - CK\_TOKEN\_INFO, [522](#)
- ulMinKeySize
  - CK\_MECHANISM\_INFO, [494](#)
- ulMinPinLen
  - CK\_TOKEN\_INFO, [523](#)
- ulNewPasswordLen
  - CK\_SKIPJACK\_RELAYX\_PARAMS, [510](#)
- ulNewPublicDataLen
  - CK\_SKIPJACK\_RELAYX\_PARAMS, [510](#)
- ulNewRandomLen
  - CK\_SKIPJACK\_RELAYX\_PARAMS, [510](#)
- ulNonceLen
  - CK\_AES\_CCM\_PARAMS, [469](#)
  - CK\_CCM\_PARAMS, [475](#)
- ulOldPasswordLen
  - CK\_SKIPJACK\_RELAYX\_PARAMS, [510](#)
- ulOldPublicDataLen
  - CK\_SKIPJACK\_RELAYX\_PARAMS, [510](#)
- ulOldRandomLen
  - CK\_SKIPJACK\_RELAYX\_PARAMS, [511](#)
- ulOldWrappedXLen
  - CK\_SKIPJACK\_RELAYX\_PARAMS, [511](#)
- ulOtherInfoLen
  - CK\_X9\_42\_DH1\_DERIVE\_PARAMS, [530](#)
  - CK\_X9\_42\_DH2\_DERIVE\_PARAMS, [531](#)
  - CK\_X9\_42\_MQV\_DERIVE\_PARAMS, [533](#)
- ulPAndGLen
  - CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS, [508](#)
- ulParameterLen
  - CK\_MECHANISM, [493](#)
- ulPasswordLen
  - CK\_PBE\_PARAMS, [496](#)
  - CK\_PKCS5\_PBKD2\_PARAMS, [498](#)
  - CK\_PKCS5\_PBKD2\_PARAMS2, [499](#)
  - CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS, [508](#)
- ulPrfDataLen
  - CK\_PKCS5\_PBKD2\_PARAMS, [498](#)
  - CK\_PKCS5\_PBKD2\_PARAMS2, [499](#)
- ulPrivateDataLen
  - CK\_ECDH2\_DERIVE\_PARAMS, [481](#)
  - CK\_ECMQV\_DERIVE\_PARAMS, [484](#)
  - CK\_X9\_42\_DH2\_DERIVE\_PARAMS, [531](#)
  - CK\_X9\_42\_MQV\_DERIVE\_PARAMS, [533](#)
- ulPublicDataLen
  - CK\_ECDH1\_DERIVE\_PARAMS, [480](#)
  - CK\_ECDH2\_DERIVE\_PARAMS, [481](#)
  - CK\_ECMQV\_DERIVE\_PARAMS, [484](#)
  - CK\_GOSTR3410\_DERIVE\_PARAMS, [487](#)
  - CK\_KEA\_DERIVE\_PARAMS, [490](#)
  - CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS, [508](#)
  - CK\_X9\_42\_DH1\_DERIVE\_PARAMS, [530](#)
  - CK\_X9\_42\_DH2\_DERIVE\_PARAMS, [531](#)
  - CK\_X9\_42\_MQV\_DERIVE\_PARAMS, [533](#)
- ulPublicDataLen2
  - CK\_ECDH2\_DERIVE\_PARAMS, [481](#)
  - CK\_ECMQV\_DERIVE\_PARAMS, [484](#)
  - CK\_X9\_42\_DH2\_DERIVE\_PARAMS, [531](#)
  - CK\_X9\_42\_MQV\_DERIVE\_PARAMS, [533](#)
- ulQLen
  - CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS, [508](#)
- ulRandomLen
  - CK\_KEA\_DERIVE\_PARAMS, [490](#)
  - CK\_SKIPJACK\_PRIVATE\_WRAP\_PARAMS, [508](#)
- ulRequestedAttributesLen
  - CK\_CMS\_SIG\_PARAMS, [477](#)
- ulRequiredAttributesLen
  - CK\_CMS\_SIG\_PARAMS, [477](#)
- ulRounds

- CK\_RC5\_CBC\_PARAMS, 501
- CK\_RC5\_MAC\_GENERAL\_PARAMS, 502
- CK\_RC5\_PARAMS, 503
- ulRwSessionCount
  - CK\_TOKEN\_INFO, 523
- ulSaltLen
  - CK\_PBE\_PARAMS, 496
- ulSaltSourceDataLen
  - CK\_PKCS5\_PBKD2\_PARAMS, 498
  - CK\_PKCS5\_PBKD2\_PARAMS2, 500
- ulSeedLen
  - CK\_DSA\_PARAMETER\_GEN\_PARAM, 479
  - CK\_KIP\_PARAMS, 492
  - CK\_TLS\_PRF\_PARAMS, 520
  - CK\_WTLS\_PRF\_PARAMS, 528
- ulSequenceNumber
  - CK\_WTLS\_KEY\_MAT\_PARAMS, 526
- ulServerOrClient
  - CK\_TLS\_MAC\_PARAMS, 519
- ulServerRandomLen
  - CK\_SSL3\_RANDOM\_DATA, 515
  - CK\_WTLS\_RANDOM\_DATA, 529
- ulSessionCount
  - CK\_TOKEN\_INFO, 523
- ulSharedDataLen
  - CK\_ECDH1\_DERIVE\_PARAMS, 480
  - CK\_ECDH2\_DERIVE\_PARAMS, 482
  - CK\_ECDH\_AES\_KEY\_WRAP\_PARAMS, 482
  - CK\_ECMQV\_DERIVE\_PARAMS, 484
- ulSourceDataLen
  - CK\_RSA\_PKCS\_OAEP\_PARAMS, 504
- ulTagBits
  - CK\_AES\_GCM\_PARAMS, 471
  - CK\_GCM\_PARAMS, 486
- ulTotalPrivateMemory
  - CK\_TOKEN\_INFO, 523
- ulTotalPublicMemory
  - CK\_TOKEN\_INFO, 523
- ulUKMLen
  - CK\_GOSTR3410\_DERIVE\_PARAMS, 487
  - CK\_GOSTR3410\_KEY\_WRAP\_PARAMS, 488
- ulValueLen
  - CK\_ATTRIBUTE, 472
  - CK\_OTP\_PARAM, 494
- ulWordsize
  - CK\_RC5\_CBC\_PARAMS, 502
  - CK\_RC5\_MAC\_GENERAL\_PARAMS, 502
  - CK\_RC5\_PARAMS, 503
- ulWrapOIDLen
  - CK\_GOSTR3410\_KEY\_WRAP\_PARAMS, 488
- ulXLen
  - CK\_KEY\_WRAP\_SET\_OAEP\_PARAMS, 491
- unlock\_mutex
  - \_pkcs11\_lib\_ctx, 400
- UnlockMutex
  - CK\_C\_INITIALIZE\_ARGS, 473
- UPDATE\_COUNT
  - calib\_command.h, 780
- update\_count
  - atca\_sign\_internal\_in\_out, 438
- UPDATE\_MODE\_DEC\_COUNTER
  - calib\_command.h, 780
- UPDATE\_MODE\_IDX
  - calib\_command.h, 781
- UPDATE\_MODE\_SELECTOR
  - calib\_command.h, 781
- UPDATE\_MODE\_USER\_EXTRA
  - calib\_command.h, 781
- UPDATE\_MODE\_USER\_EXTRA\_ADD
  - calib\_command.h, 781
- UPDATE\_RSP\_SIZE
  - calib\_command.h, 781
- UPDATE\_VALUE\_IDX
  - calib\_command.h, 781
- USART\_BAUD\_RATE
  - swi\_uart\_start.c, 1082
- usart\_instance
  - atcaSWImaster, 467
- USART\_SWI
  - atcaSWImaster, 467
- use\_flag
  - atca\_sign\_internal\_in\_out, 438
- UseLock
  - \_atecc608\_config, 394
- user\_pin\_handle
  - \_pkcs11\_slot\_ctx, 407
- UserExtra
  - \_atecc508a\_config, 390
  - \_atecc608\_config, 394
  - \_atsha204a\_config, 397
- UserExtraAdd
  - \_atecc608\_config, 394
- utcTime
  - CK\_TOKEN\_INFO, 523
- valid
  - atca\_temp\_key, 440
- value
  - atca\_temp\_key, 440
- VERIFY\_256\_EXTERNAL\_COUNT
  - calib\_command.h, 782
- VERIFY\_256\_KEY\_SIZE
  - calib\_command.h, 782
- VERIFY\_256\_SIGNATURE\_SIZE
  - calib\_command.h, 782
- VERIFY\_256\_STORED\_COUNT
  - calib\_command.h, 782
- VERIFY\_256\_VALIDATE\_COUNT
  - calib\_command.h, 782
- VERIFY\_283\_EXTERNAL\_COUNT
  - calib\_command.h, 782
- VERIFY\_283\_KEY\_SIZE
  - calib\_command.h, 783
- VERIFY\_283\_SIGNATURE\_SIZE
  - calib\_command.h, 783
- VERIFY\_283\_STORED\_COUNT
  - calib\_command.h, 783

VERIFY\_283\_VALIDATE\_COUNT  
     calib\_command.h, [783](#)  
 VERIFY\_DATA\_IDX  
     calib\_command.h, [783](#)  
 VERIFY\_KEY\_B283  
     calib\_command.h, [783](#)  
 VERIFY\_KEY\_K283  
     calib\_command.h, [784](#)  
 VERIFY\_KEY\_P256  
     calib\_command.h, [784](#)  
 VERIFY\_KEYID\_IDX  
     calib\_command.h, [784](#)  
 VERIFY\_MODE\_EXTERNAL  
     calib\_command.h, [784](#)  
 VERIFY\_MODE\_IDX  
     calib\_command.h, [784](#)  
 VERIFY\_MODE\_INVALIDATE  
     calib\_command.h, [784](#)  
 VERIFY\_MODE\_MAC\_FLAG  
     calib\_command.h, [785](#)  
 VERIFY\_MODE\_MASK  
     calib\_command.h, [785](#)  
 VERIFY\_MODE\_SOURCE\_MASK  
     calib\_command.h, [785](#)  
 VERIFY\_MODE\_SOURCE\_MSGDIGBUF  
     calib\_command.h, [785](#)  
 VERIFY\_MODE\_SOURCE\_TEMPKEY  
     calib\_command.h, [785](#)  
 VERIFY\_MODE\_STORED  
     calib\_command.h, [785](#)  
 VERIFY\_MODE\_VALIDATE  
     calib\_command.h, [786](#)  
 VERIFY\_MODE\_VALIDATE\_EXTERNAL  
     calib\_command.h, [786](#)  
 verify\_other\_data  
     atca\_sign\_internal\_in\_out, [438](#)  
 VERIFY\_OTHER\_DATA\_SIZE  
     calib\_command.h, [786](#)  
 VERIFY\_RSP\_SIZE  
     calib\_command.h, [786](#)  
 VERIFY\_RSP\_SIZE\_MAC  
     calib\_command.h, [786](#)  
 version  
     CK\_FUNCTION\_LIST, [485](#)  
 version\_info  
     memory\_parameters, [538](#)  
 vid  
     ATCAIfaceCfg, [464](#)  
 VolatileKeyPermission  
     \_atecc608\_config, [394](#)  
  
 wake\_delay  
     ATCAIfaceCfg, [464](#)  
 WARRANTIES  
     license.txt, [893](#)  
 WARRANTY  
     license.txt, [894](#)  
 wordsize  
     ATCAIfaceCfg, [464](#)  
  
 WRITE\_ADDR\_IDX  
     calib\_command.h, [786](#)  
 write\_handle  
     hid\_device, [535](#)  
 WRITE\_MAC\_SIZE  
     calib\_command.h, [787](#)  
 WRITE\_MAC\_VL\_IDX  
     calib\_command.h, [787](#)  
 WRITE\_MAC\_VS\_IDX  
     calib\_command.h, [787](#)  
 WRITE\_RSP\_SIZE  
     calib\_command.h, [787](#)  
 WRITE\_VALUE\_IDX  
     calib\_command.h, [787](#)  
 WRITE\_ZONE\_DATA  
     calib\_command.h, [787](#)  
 WRITE\_ZONE\_IDX  
     calib\_command.h, [788](#)  
 WRITE\_ZONE\_MASK  
     calib\_command.h, [788](#)  
 WRITE\_ZONE\_OTP  
     calib\_command.h, [788](#)  
 WRITE\_ZONE\_WITH\_MAC  
     calib\_command.h, [788](#)  
  
 X509format  
     \_atecc508a\_config, [390](#)  
     \_atecc608\_config, [395](#)  
  
 y  
     atca\_aes\_gcm\_ctx, [413](#)  
 year  
     CK\_DATE, [478](#)  
  
 zero  
     Host side crypto methods (atcah\_), [330](#)  
 zone  
     atca\_gen\_dig\_in\_out, [423](#)  
     atca\_write\_mac\_in\_out, [445](#)  
     atcacert\_device\_loc\_s, [454](#)