Bluetooth® Low Energy Host Stack API Reference Manual

BLEHSAPIRM

Rev. 13
May 2020

Contents

Chapter 1 BLE Configuration Constants

Chapter 5 GATT - Client APIs

Chapter 1
BLE Configuration Constants

1.1 Overview

Files

- file ble_constants.h

Macros

- #define gcBleDeviceAddressSize_c
- #define gBleBondIdentityHeaderSize_c
- #define **gBleBondDataDynamicSize_c**
- #define **gBleBondDataStaticSize_c**
- #define **gBleBondDataDeviceInfoSize_c**
- #define **gBleBondDataDescriptorSize_c**
- #define gcGapMaximumSavedCccds_c
- #define gcGapMaxAuthorizationHandles_c
- #define gBleBondDataSize_c
- #define gcGapMaxServiceSpecificSecurityRequirements_c
- #define gcBleLongUuidSize_c
- #define gcSmpMaxLtkSize_c
- #define gcSmpIrkSize_c
- #define gcSmpCsrkSize_c
- #define gcSmpMaxRandSize_c
- #define gcSmpOobSize_c
- #define gSmpLeScRandomValueSize_c
- #define gSmpLeScRandomConfirmValueSize_c
- #define gcGapMaxDeviceNameSize_c
- #define gcGapMaxAdvertisingDataLength_c
- #define gAttDefaultMtu_c
- #define gAttMaxMtu_c
- #define gAttMaxValueLength_c
- #define gHciTransportUartChannel_c
- #define gcReservedFlashSizeForCustomInformation_c
- #define gcBleChannelMapSize_c
- #define **gBleMinTxOctets_c**
- #define **gBleMinTxTime_c**
- #define **gBleMaxTxOctets_c**
- #define **gBleMaxTxTime_c**
- #define **gBleMaxTxTimeCodedPhy_c**
- #define gBleExtAdvMaxSetId_c
- #define gBlePeriodicAdvMaxSyncHandle_c
- #define gBleExtAdvLegacySetId_c
- #define gBleExtAdvLegacySetHandle_c
- #define gBleExtAdvDefaultSetId_c
- #define gBleExtAdvDefaultSetHandle_c
- #define gBleAdvTxPowerNoPreference_c

- #define gBleExtAdvNoDuration_c
- #define gBleHighDutyDirectedAdvDuration
- #define gBleExtAdvNoMaxEvents_c
- #define gBlePeriodicAdvDefaultHandle_c
- #define gBlePeriodicAdvSyncTimeoutMin_c
- #define gBlePeriodicAdvSyncTimeoutMax_c
- #define gBlePeriodicAdvSkipMax_c
- #define gBleMaxADStructureLength_c
- #define gBleMaxExtAdvDataLength_c
- #define gBleExtAdvMaxAuxOffsetUsec_c

1.2 Macro Definition Documentation

1.2.1 #define gcBleDeviceAddressSize_c

Size of a BLE Device Address.

1.2.2 #define gBleBondIdentityHeaderSize_c

Size of bond data structures for a bonded device.

1.2.3 #define gcGapMaximumSavedCccds_c

Maximum number of CCCDs.

1.2.4 #define gcGapMaxAuthorizationHandles_c

Maximum number of attributes that require authorization.

1.2.5 #define gBleBondDataSize_c

Bonding Data Size.

1.2.6 #define gcGapMaxServiceSpecificSecurityRequirements_c

Maximum number of gapServiceSecurityRequirements_t structures that can be registered with Gap_↩ RegisterDeviceSecurityRequirements()

1.2.7 #define gcBleLongUuidSize_c

Size of long UUIDs.

1.2.8 #define gcSmpMaxLtkSize_c

Maximum Long Term Key size in bytes.

1.2.9 #define gcSmpIrkSize_c

Identity Resolving Key size in bytes.

1.2.10 #define gcSmpCsrkSize_c

Connection Signature Resolving Key size in bytes.

1.2.11 #define gcSmpMaxRandSize_c

Maximum Rand size in bytes.

1.2.12 #define gcSmpOobSize_c

SMP OOB size in bytes.

1.2.13 #define gSmpLeScRandomValueSize_c

SMP LE Secure Connections Pairing Random size in bytes.

1.2.14 #define gSmpLeScRandomConfirmValueSize_c

SMP LE Secure Connections Pairing Confirm size in bytes.

1.2.15 #define gcGapMaxDeviceNameSize_c

Maximum device name size.

1.2.16 #define gcGapMaxAdvertisingDataLength_c

Maximum size of advertising and scan response data.

1.2.17 #define gAttDefaultMtu_c

Default value of the ATT_MTU.

1.2.18 #define gAttMaxMtu_c

Maximum possible value of the ATT_MTU for this device.

This is used during the MTU Exchange.

1.2.19 #define gAttMaxValueLength_c

The maximum length of an attribute value shall be 512 octets.

1.2.20 #define gHciTransportUartChannel_c

Channel the number of the UART hardware module (For example, if UART1 is used, this value should be 1).

1.2.21 #define gcReservedFlashSizeForCustomInformation_c

Number of bytes reserved for storing application-specific information about a device.

1.2.22 #define gcBleChannelMapSize_c

Size of a channel map in a connection.

1.2.23 #define gBleExtAdvMaxSetId_c

Maximum value of the advertising SID.

1.2.24 #define gBlePeriodicAdvMaxSyncHandle_c

Maximum value of the periodic advertising handle.

1.2.25 #define gBleExtAdvLegacySetId_c

SID of the legacy advertising set.

1.2.26 #define gBleExtAdvLegacySetHandle_c

Handle of the legacy advertising set.

1.2.27 #define gBleExtAdvDefaultSetId_c

Default SID for extended advertising.

1.2.28 #define gBleExtAdvDefaultSetHandle_c

Default handle for extended advertising.

1.2.29 #define gBleAdvTxPowerNoPreference_c

Host has no preference for Tx Power.

1.2.30 #define gBleExtAdvNoDuration_c

No advertising duration.

Advertising to continue until the Host disables it.

1.2.31 #define gBleHighDutyDirectedAdvDuration

Default advertising duration in high duty directed advertising 1.28s = 1280ms/10ms(unit) = 128.

1.2.32 #define gBleExtAdvNoMaxEvents_c

No maximum number of advertising events.

1.2.33 #define gBlePeriodicAdvDefaultHandle_c

Periodic advertising default handle.

1.2.34 #define gBlePeriodicAdvSyncTimeoutMin_c

Minimum value for the sync_timeout parameter.

1.2.35 #define gBlePeriodicAdvSyncTimeoutMax_c

Maximum value for the sync_timeout parameter.

1.2.36 #define gBlePeriodicAdvSkipMax_c

Maximum value for the skip parameter.

1.2.37 #define gBleMaxADStructureLength_c

Maximum length of an AD structure.

1.2.38 #define gBleMaxExtAdvDataLength_c

Maximum length of Extended Advertising Data.

1.2.39 #define gBleExtAdvMaxAuxOffsetUsec_c

Maximum value in us of AUX Offset(13 bits) in AuxPtr in 300us units, i.e.
((1 << 13) - 1) ∗ 300

Chapter 2
BLE General Definitions

2.1 Overview

Files

- file ble_general.h
- file ble_host_tasks.h
- file ble_sig_defines.h
- file ble_utils.h

Data Structures

- struct bleIdentityAddress_t
- union bleUuid_t
- struct bleAdvertisingChannelMap_t
- struct gapLeScOobData_t
- struct gapInternalError_t
- struct gapControllerTestEvent_t
- struct gapPhyEvent_t
- struct bleNotificationEvent_t
- struct gapInitComplete_t
- struct bleBondCreatedEvent_t
- struct gapAddrReadyEvent_t
- struct gapGenericEvent_t
- union gapGenericEvent_t.eventData
- struct bleBondIdentityHeaderBlob_t
- struct bleBondDataDynamicBlob_t
- struct bleBondDataStaticBlob_t
- struct bleBondDataDeviceInfoBlob_t
- struct bleBondDataDescriptorBlob_t
- struct bleBondDataBlob_t

Macros

- #define **gInvalidDeviceId_c**
- #define **gInvalidNvmIndex_c**
- #define gcConnectionIntervalMin_c
- #define **gcConnectionIntervalMax_c**
- #define **gcConnectionSlaveLatencyMax_c**
- #define **gcConnectionSupervisionTimeoutMin_c**
- #define **gcConnectionSupervisionTimeoutMax_c**
- #define gcConnectionIntervalMinDefault_c
- #define gcConnectionIntervalMaxDefault_c
- #define **gcConnectionSlaveLatencyDefault_c**
- #define gcConnectionSupervisionTimeoutDefault_c
- #define gcConnectionEventMinDefault_c

- #define gcConnectionEventMaxDefault_c
- #define STATIC
- #define gBleAddrTypePublic_c
- #define gBleAddrTypeRandom_c
- #define Ble_IsPrivateResolvableDeviceAddress(bleAddress)
- #define Ble_IsPrivateNonresolvableDeviceAddress(bleAddress)
- #define Ble_IsRandomStaticDeviceAddress(bleAddress)
- #define Ble_DeviceAddressesMatch(bleAddress1, bleAddress2)
- #define Ble_CopyDeviceAddress(destinationAddress, sourceAddress)
- #define gBleUuidType16_c
- #define gBleUuidType128_c
- #define gBleUuidType32_c
- #define gLePhy1MFlag_c
- #define gLePhy2MFlag_c
- #define gLePhyCodedFlag_c
- #define gUseDeviceAddress_c
- #define gUseWhiteList_c
- #define gScanAll_c
- #define gScanWithWhiteList_c
- #define gNetworkPrivacy_c
- #define gDevicePrivacy_c
- #define gBleSig_PrimaryService_d
- #define gBleSig_SecondaryService_d
- #define gBleSig_Include_d
- #define gBleSig_Characteristic_d
- #define gBleSig_CCCD_d
- #define gBleSig_SCCD_d
- #define gBleSig_CharPresFormatDescriptor_d
- #define gBleSig_ValidRangeDescriptor_d
- #define gBleSig_GenericAccessProfile_d
- #define gBleSig_GenericAttributeProfile_d
- #define gBleSig_ImmediateAlertService_d
- #define gBleSig_LinkLossService_d
- #define gBleSig_TxPowerService_d
- #define gBleSig_CurrentTimeService_d
- #define gBleSig_ReferenceTimeUpdateService_d
- #define gBleSig_NextDSTChangeService_d
- #define gBleSig_GlucoseService_d
- #define gBleSig_HealthThermometerService_d
- #define gBleSig_DeviceInformationService_d
- #define gBleSig_HeartRateService_d
- #define gBleSig_PhoneAlertStatusService_d
- #define gBleSig_BatteryService_d
- #define gBleSig_BloodPressureService_d
- #define gBleSig_AlertNotificationService_d
- #define gBleSig_HidService_d
- #define gBleSig_RunningSpeedAndCadenceService_d
- #define gBleSig_CyclingSpeedAndCadenceService_d
- #define gBleSig_CyclingPowerService_d
- #define gBleSig_LocationAndNavigationService_d
- #define gBleSig_IpsService_d
- #define gBleSig_PulseOximeterService_d
- #define gBleSig_HTTPProxyService_d
- #define gBleSig_WPTService_d
- #define gBleSig_BtpService_d
- #define gBleSig_GapDeviceName_d

- #define gBleSig_GapAppearance_d
- #define gBleSig_GapPpcp_d
- #define gBleSig_GattServiceChanged_d
- #define gBleSig_AlertLevel_d
- #define gBleSig_TxPower_d
- #define gBleSig_LocalTimeInformation_d
- #define gBleSig_TimeWithDST_d
- #define gBleSig_ReferenceTimeInformation_d
- #define gBleSig_TimeUpdateControlPoint_d
- #define gBleSig_TimeUpdateState_d
- #define gBleSig_GlucoseMeasurement_d
- #define gBleSig_BatteryLevel_d
- #define gBleSig_TemperatureMeasurement_d
- #define gBleSig_TemperatureType_d
- #define gBleSig_IntermediateTemperature_d
- #define gBleSig_MeasurementInterval_d
- #define gBleSig_SystemId_d
- #define gBleSig_ModelNumberString_d
- #define gBleSig_SerialNumberString_d
- #define gBleSig_FirmwareRevisionString_d
- #define gBleSig_HardwareRevisionString_d
- #define gBleSig_SoftwareRevisionString_d
- #define gBleSig_ManufacturerNameString_d
- #define gBleSig_IeeeRcdl_d
- #define gBleSig_CurrentTime_d
- #define gBleSig_BootKeyboardInputReport_d
- #define gBleSig_BootKeyboardOutputReport_d
- #define gBleSig_BootMouseInputReport_d
- #define gBleSig_GlucoseMeasurementContext_d
- #define gBleSig_BpMeasurement_d
- #define gBleSig_IntermediateCuffPressure_d
- #define gBleSig_HrMeasurement_d
- #define gBleSig_BodySensorLocation_d
- #define gBleSig_HrControlPoint_d
- #define gBleSig_AlertStatus_d
- #define gBleSig_RingerControlPoint_d
- #define gBleSig_RingerSetting_d
- #define gBleSig_AlertNotifControlPoint_d
- #define gBleSig_UnreadAlertStatus_d
- #define gBleSig_NewAlert_d
- #define gBleSig_SupportedNewAlertCategory_d
- #define gBleSig_SupportedUnreadAlertCategory_d
- #define gBleSig_BloodPressureFeature_d
- #define gBleSig_HidInformation_d
- #define gBleSig_HidCtrlPoint_d
- #define gBleSig_Report_d
- #define gBleSig_ProtocolMode_d
- #define gBleSig_ScanIntervalWindow_d
- #define gBleSig_PnpId_d
- #define gBleSig_GlucoseFeature_d
- #define gBleSig_RaCtrlPoint_d
- #define gBleSig_RscMeasurement_d
- #define gBleSig_RscFeature_d
- #define gBleSig_ScControlPoint_d
- #define gBleSig_CscMeasurement_d
- #define gBleSig_CscFeature_d

- #define gBleSig_SensorLocation_d
- #define gBleSig_PlxSCMeasurement_d
- #define gBleSig_PlxContMeasurement_d
- #define gBleSig_PulseOximeterFeature_d
- #define gBleSig_CpMeasurement_d
- #define gBleSig_CpVector_d
- #define gBleSig_CpFeature_d
- #define gBleSig_CpControlPoint_d
- #define gBleSig_LocationAndSpeed_d
- #define gBleSig_Navigation_d
- #define gBleSig_PositionQuality_d
- #define gBleSig_LnFeature_d
- #define gBleSig_LnControlPoint_d
- #define gBleSig_Temperature_d
- #define gBleSig_CentralAddressResolution_d
- #define gBleSig_URI_d
- #define gBleSig_HTTP_Headers_d
- #define gBleSig_HTTP_StatusCode_d
- #define gBleSig_HTTP_EntityBody_d
- #define gBleSig_HTTP_ControlPoint_d
- #define gBleSig_HTTPS_Security_d
- #define gBleSig_ResolvablePrivateAddressOnly_d
- #define gBleSig_MeshProvisioningService_d
- #define gBleSig_MeshProxyService_d
- #define gBleSig_MeshProvDataIn_d
- #define gBleSig_MeshProvDataOut_d
- #define gBleSig_MeshProxyDataIn_d
- #define gBleSig_MeshProxyDataOut_d
- #define gBleSig_CAR_NotSupported_d
- #define **gBleSig_CAR_Supported_d**
- #define gBleSig_RPAO_Used_d
- #define BleSig_IsGroupingAttributeUuid16(uuid16)
- #define BleSig_IsServiceDeclarationUuid16(uuid16)
- #define Uuid16(uuid)
- #define Uuid32(uuid)
- #define **UuidArray**(value)
- #define PACKED_STRUCT
- #define global
- #define __noreturn
- #define Utils_ExtractTwoByteValue(buf)
- #define Utils_ExtractThreeByteValue(buf)
- #define Utils_ExtractFourByteValue(buf)
- #define Utils_BeExtractTwoByteValue(buf)
- #define Utils_BeExtractThreeByteValue(buf)
- #define Utils_BeExtractFourByteValue(buf)
- #define Utils_PackTwoByteValue(value, buf)
- #define Utils_PackThreeByteValue(value, buf)
- #define Utils_PackFourByteValue(value, buf)
- #define Utils_BePackTwoByteValue(value, buf)
- #define Utils_BePackThreeByteValue(value, buf)
- #define Utils_BePackFourByteValue(value, buf)
- #define Utils_Copy8(ptr, val8)
- #define Utils_Copy16(ptr, val16)
- #define Utils_Copy32(ptr, val32)
- #define Utils_Copy64(ptr, val64)
- #define Utils_RevertByteArray(array, size)

Typedefs

- typedef uint8_t deviceId_t
- typedef uint8_t bleAddressType_t
- typedef uint8_t bleDeviceAddress_t[gcBleDeviceAddressSize_c]
- typedef uint8_t bleUuidType_t
- typedef uint16_t bleAdvReportEventProperties_t
- typedef uint16_t bleAdvRequestProperties_t
- typedef uint8_t **bleMasterClockAccuracy_t**
- typedef uint8_t bleScanningFilterPolicy_t
- typedef uint8_t bleInitiatorFilterPolicy_t
- typedef uint8_t blePrivacyMode_t
- typedef uint8_t bleChannelMap_t[gcBleChannelMapSize_c]
- typedef uint8_t gapLePhyFlags_t
- typedef uint8_t gapLePhyMode_t
- typedef uint16_t bleNotificationEventType_t
- typedef void(∗ gapGenericCallback_t) (gapGenericEvent_t ∗pGenericEvent)
- typedef bleResult_t(∗ hciHostToControllerInterface_t) (hciPacketType_t packetType, void ∗p↩ Packet, uint16_t packetSize)
- typedef uint32_t **LeSupportedFeatures_t**

Enumerations

- enum bleResult_t {
  gBleStatusBase_c,
  gBleSuccess_c,
  gBleInvalidParameter_c,
  gBleOverflow_c,
  gBleUnavailable_c,
  gBleFeatureNotSupported_c,
  gBleOutOfMemory_c,
  gBleAlreadyInitialized_c,
  gBleOsError_c,
  gBleUnexpectedError_c,
  gBleInvalidState_c,
  gBleTimerError_c,
  gHciStatusBase_c,
  gHciSuccess_c,
  gHciUnknownHciCommand_c,
  gHciUnknownConnectionIdentifier_c,
  gHciHardwareFailure_c,
  gHciPageTimeout_c,
  gHciAuthenticationFailure_c,
  gHciPinOrKeyMissing_c,
  gHciMemoryCapacityExceeded_c,
  gHciConnectionTimeout_c,
  gHciConnectionLimitExceeded_c,
  gHciSynchronousConnectionLimitToADeviceExceeded_c,
  gHciAclConnectionAlreadyExists_c,
  gHciCommandDisallowed_c,
  gHciConnectionRejectedDueToLimitedResources_c,
  gHciConnectionRejectedDueToSecurityReasons_c,
  gHciConnectionRejectedDueToUnacceptableBdAddr_c,
  gHciConnectionAcceptTimeoutExceeded_c,
  gHciUnsupportedFeatureOrParameterValue_c,
  gHciInvalidHciCommandParameters_c,
  gHciRemoteUserTerminatedConnection_c,
  gHciRemoteDeviceTerminatedConnectionLowResources_c,
  gHciRemoteDeviceTerminatedConnectionPowerOff_c,
  gHciConnectionTerminatedByLocalHost_c,
  gHciRepeatedAttempts_c,
  gHciPairingNotAllowed_c,
  gHciUnknownLpmPdu_c,
  gHciUnsupportedRemoteFeature_c,
  gHciScoOffsetRejected_c,
  gHciScoIntervalRejected_c,
  gHciScoAirModeRejected_c,
  gHciInvalidLpmParameters_c,
  gHciUnspecifiedError_c,
  gHciUnsupportedLpmParameterValue_c,

gGattDbDescriptorNotFound_c }
- enum bleAdvertisingType_t {
  gAdvConnectableUndirected_c,
  gAdvDirectedHighDutyCycle_c,
  gAdvScannable_c,
  gAdvNonConnectable_c,
  gAdvDirectedLowDutyCycle_c }
- enum bleAdvReportEventProperties_tag {
  gAdvEventConnectable_c,
  gAdvEventScannable_c,
  gAdvEventDirected_c,
  gAdvEventScanResponse_c,
  gAdvEventLegacy_c,
  gAdvEventAnonymous_c }
- enum bleAdvRequestProperties_tag {
  gAdvReqConnectable_c,
  gAdvReqScannable_c,
  gAdvReqDirected_c,
  gAdvReqHighDutyCycle_c,
  gAdvReqLegacy_c,
  gAdvReqAnonymous_c,
  gAdvIncludeTxPower_c }
- enum bleAdvertisingFilterPolicy_t {
  gBleAdvFilterAllowScanFromAnyAllowConnFromAny_c,
  gBleAdvFilterAllowScanFromWLAllowConnFromAny_c,
  gBleAdvFilterAllowScanFromAnyAllowConnFromWL_c,
  gBleAdvFilterAllowScanFromWLAllowConnFromWL_c }
- enum bleLlConnectionRole_t {
  gBleLlConnectionMaster_c,
  gBleLlConnectionSlave_c }
- enum **bleMasterClockAccuracy_tag** {
  gBleMasterClkAcc500ppm_c,
  gBleMasterClkAcc250ppm_c,
  gBleMasterClkAcc150ppm_c,
  gBleMasterClkAcc100ppm_c,
  gBleMasterClkAcc75ppm_c,
  gBleMasterClkAcc50ppm_c,
  gBleMasterClkAcc30ppm_c,
  gBleMasterClkAcc20ppm_c }
- enum **bleAdvertisingReportEventType_t** {
  gBleAdvRepAdvInd_c,
  gBleAdvRepAdvDirectInd_c,
  gBleAdvRepAdvScanInd_c,
  gBleAdvRepAdvNonconnInd_c,
  gBleAdvRepScanRsp_c }
- enum hciPacketType_t {

gHciCommandPacket_c,
gHciDataPacket_c,
gHciSynchronousDataPacket_c,
gHciEventPacket_c }
- enum bleScanType_t {
gScanTypePassive_c,
gScanTypeActive_c }
- enum bleTransmitPowerLevelType_t {
gReadCurrentTxPowerLevel_c,
gReadMaximumTxPowerLevel_c }
- enum bleTransmitPowerChannelType_t {
gTxPowerAdvChannel_c,
gTxPowerConnChannel_c }
- enum **bleChannelFrequency_t** {

gBleFreq2402MHz_c,
gBleFreq2404MHz_c,
gBleFreq2406MHz_c,
gBleFreq2408MHz_c,
gBleFreq2410MHz_c,
gBleFreq2412MHz_c,
gBleFreq2414MHz_c,
gBleFreq2416MHz_c,
gBleFreq2418MHz_c,
gBleFreq2420MHz_c,
gBleFreq2422MHz_c,
gBleFreq2424MHz_c,
gBleFreq2426MHz_c,
gBleFreq2428MHz_c,
gBleFreq2430MHz_c,
gBleFreq2432MHz_c,
gBleFreq2434MHz_c,
gBleFreq2436MHz_c,
gBleFreq2438MHz_c,
gBleFreq2440MHz_c,
gBleFreq2442MHz_c,
gBleFreq2444MHz_c,
gBleFreq2446MHz_c,
gBleFreq2448MHz_c,
gBleFreq2450MHz_c,
gBleFreq2452MHz_c,
gBleFreq2454MHz_c,
gBleFreq2456MHz_c,
gBleFreq2458MHz_c,
gBleFreq2460MHz_c,
gBleFreq2462MHz_c,
gBleFreq2464MHz_c,
gBleFreq2466MHz_c,
gBleFreq2468MHz_c,
gBleFreq2470MHz_c,
gBleFreq2472MHz_c,
gBleFreq2474MHz_c,
gBleFreq2476MHz_c,
gBleFreq2478MHz_c,
gBleFreq2480MHz_c }
- enum **bleTxTestPacketPayload_t** {

gBleTestPacketPayloadPrbs9_c,
gBleTestPacketPayloadPattern11110000_c,
gBleTestPacketPayloadPattern10101010_c,
gBleTestPacketPayloadPrbs15_c,
gBleTestPacketPayloadPatternAllBits1_c,
gBleTestPacketPayloadPatternAllBits0_c,
gBleTestPacketPayloadPattern00001111_c,
gBleTestPacketPayloadPattern01010101_c }
- enum **bleHardwareErrorCode_t** { **bleHwErrCodeNoError_c** }
- enum gapGenericEventType_t {
gInitializationComplete_c,
gInternalError_c,
gAdvertisingSetupFailed_c,
gAdvertisingParametersSetupComplete_c,
gAdvertisingDataSetupComplete_c,
gWhiteListSizeRead_c,
gDeviceAddedToWhiteList_c,
gDeviceRemovedFromWhiteList_c,
gWhiteListCleared_c,
gRandomAddressReady_c,
gCreateConnectionCanceled_c,
gPublicAddressRead_c,
gAdvTxPowerLevelRead_c,
gPrivateResolvableAddressVerified_c,
gRandomAddressSet_c,
gLeScPublicKeyRegenerated_c,
gLeScLocalOobData_c,
gHostPrivacyStateChanged_c,
gControllerPrivacyStateChanged_c,
gControllerTestEvent_c,
gTxPowerLevelSetComplete_c,
gLePhyEvent_c,
gControllerNotificationEvent_c,
gBondCreatedEvent_c,
gChannelMapSet_c,
gExtAdvertisingParametersSetupComplete_c,
gExtAdvertisingDataSetupComplete_c,
gExtAdvertisingSetRemoveComplete_c,
gPeriodicAdvParamSetupComplete_c,
gPeriodicAdvDataSetupComplete_c,
gPeriodicAdvListUpdateComplete_c,
gPeriodicAdvCreateSyncCancelled_c,
gTxEntryAvailable_c }
- enum gapInternalErrorSource_t {

gHciCommandStatus_c,
gCheckPrivateResolvableAddress_c,
gVerifySignature_c,
gAddNewConnection_c,
gResetController_c,
gSetEventMask_c,
gReadLeBufferSize_c,
gSetLeEventMask_c,
gReadDeviceAddress_c,
gReadLocalSupportedFeatures_c,
gReadWhiteListSize_c,
gClearWhiteList_c,
gAddDeviceToWhiteList_c,
gRemoveDeviceFromWhiteList_c,
gCancelCreateConnection_c,
gReadRadioPower_c,
gSetRandomAddress_c,
gCreateRandomAddress_c,
gEncryptLink_c,
gProvideLongTermKey_c,
gDenyLongTermKey_c,
gConnect_c,
gDisconnect_c,
gTerminatePairing_c,
gSendSlaveSecurityRequest_c,
gEnterPasskey_c,
gProvideOob_c,
gSendSmpKeys_c,
gWriteSuggestedDefaultDataLength_c,
gReadSuggestedDefaultDataLength_c,
gUpdateLeDataLength_c,
gEnableHostPrivacy_c,
gEnableControllerPrivacy_c,
gLeScSendKeypressNotification_c,
gLeScSetPeerOobData_c,
gLeScGetLocalOobData_c,
gLeScValidateNumericValue_c,
gLeScRegeneratePublicKey_c,
gLeSetResolvablePrivateAddressTimeout_c,
gDefaultPairingProcedure_c,
gLeControllerTest_c,
gLeReadPhy_c,
gLeSetPhy_c,
gSaveKeys_c,
gSetChannelMap_c,
gReadLocalSupportedCommands_c,
gEnableLdmTimer_c,
gRemoveAdvertisingSet_c,
gLePeriodicAdvSyncEstb_c

gExtAdvReportProcess_c }
- enum gapControllerTestEventType_t {
  gControllerReceiverTestStarted_c,
  gControllerTransmitterTestStarted_c,
  gControllerTestEnded_c }
- enum gapLeAllPhyFlags_t {
  gLeTxPhyNoPreference_c,
  gLeRxPhyNoPreference_c }
- enum gapLePhyOptionsFlags_t {
  gLeCodingNoPreference_c,
  gLeCodingS2_c,
  gLeCodingS8_c }
- enum gapLePhyMode_tag {
  gLePhy1M_c,
  gLePhy2M_c,
  gLePhyCoded_c }
- enum gapPhyEventType_t {
  gPhySetDefaultComplete_c,
  gPhyRead_c,
  gPhyUpdateComplete_c }
- enum bleNotificationEventType_tag {
  gNotifEventNone_c,
  gNotifConnEventOver_c,
  gNotifConnRxPdu_c,
  gNotifAdvEventOver_c,
  gNotifAdvTx_c,
  gNotifAdvScanReqRx_c,
  gNotifAdvConnReqRx_c,
  gNotifScanEventOver_c,
  gNotifScanAdvPktRx_c,
  gNotifScanRspRx_c,
  gNotifScanReqTx_c,
  gNotifConnCreated_c }
- enum **LeSupportedFeatures_tag** {

gLeEncryption_c,
gLeConnectionParametersRequestProcedure_c,
gLeExtendedRejectIndication_c,
gLeSlaveInitiatedFeaturesExchange_c,
gLePing_c,
gLeDataPacketLengthExtension_c,
gLeLlPrivacy_c,
gLeExtendedScannerFilterPolicies_c,
gLe2MbPhy_c,
gLeStableModulationIdxTx_c,
gLeStableModulationIdxRx_c,
gLeCodedPhy_c,
gLeExtendedAdv_c,
gLePeriodicAdv_c,
gLeChannelSelAlg2_c,
gLePowerClass1_c,
gLeMinNumOfUsedChanProcedure_c }

Functions

- bleResult_t Ble_HostInitialize (gapGenericCallback_t genericCallback, hciHostToControllerInterface_t hostToControllerInterface)
- bleResult_t Ble_HciRecv (hciPacketType_t packetType, void ∗pHciPacket, uint16_t packetSize)
- void Host_TaskHandler (void ∗args)

Variables

- const uint8_t **gBleMaxActiveConnections**
- msgQueue_t gApp2Host_TaskQueue
- msgQueue_t gHci2Host_TaskQueue
- osaEventId_t gHost_TaskEvent

2.2 Data Structure Documentation

2.2.1 struct bleIdentityAddress_t

Bluetooth Identity Address - array of 6 bytes.

Data Fields

bleAddressType_t	idAddressType	Public or Random (static).

bleDevice↩ Address_t	idAddress	6-byte address.

2.2.2 union bleUuid_t

Union for a Bluetooth UUID; selected according to an accompanying bleUuidType_t.

Data Fields

uint16_t	uuid16	For gBleUuidType16_c.
uint32_t	uuid32	For gBleUuidType32_c.
uint8_t	uuid128[16]	For gBleUuidType128_c.

2.2.3 struct bleAdvertisingChannelMap_t

Data Fields

uint8_t	enable↩ Channel37: 1	Bit for channel 37.
uint8_t	enable↩ Channel38: 1	Bit for channel 38.
uint8_t	enable↩ Channel39: 1	Bit for channel 39.
uint8_t	reserved: 5	Reserved for future use.

2.2.4 struct gapLeScOobData_t

Data Fields

uint8_t	random↩ Value[gSmp↩ LeScRandom↩ ValueSize_c]	LE SC OOB r (Random value)

uint8_t	confirm↩ Value[gSmp↩ LeScRandom↩ Confirm↩ ValueSize_c]	LE SC OOB Cr (Random Confirm value)

2.2.5 struct gapInternalError_t

Internal Error Event Data.

Data Fields

bleResult_t	errorCode	Host Stack error code.
gapInternal↩ ErrorSource_t	errorSource	The command that generated the error; useful when it is not obvious from the error code.
uint16_t	hciCommand↩ Opcode	Only for errorSource = gHciCommandStatus_c; the HCI Command that received an error status.

2.2.6 struct gapControllerTestEvent_t

Controller Test Event.

Data Fields

gapController↩ TestEvent↩ Type_t	testEventType	
uint16_t	received↩ Packets	

2.2.7 struct gapPhyEvent_t

Phy Event.

Data Fields

gapPhyEvent↩ Type_t	phyEventType	
deviceId_t	deviceId	
uint8_t	txPhy	

uint8_t	rxPhy	

2.2.8 struct bleNotificationEvent_t

Controller Enhanced Notification Event.

Data Fields

ble↩ Notification↩ EventType_t	eventType	Enhanced notification event type.
deviceId_t	deviceId	Device id of the peer, valid for connection events.
int8_t	rssi	RSSI, valid for Rx event types.
uint8_t	channel	Channel, valid for conn event over or Rx/Tx events.
uint16_t	ce_counter	Connection event counter, valid for conn event over or Conn Rx event.
bleResult_t	status	Status of the request to select which events to be enabled/disabled.
uint16_t	timestamp	Timestamp in 625 us slots, valid for Conn Rx event and Conn Created event.
uint8_t	adv_handle	Advertising Handle, valid for advertising events, if multiple ADV sets supported.

2.2.9 struct gapInitComplete_t

gInitializationComplete_c event data

Data Fields

uint32_t	supported↩ Features	
uint16_t	maxAdvData↩ Size	
uint8_t	numOf↩ Supported↩ AdvSets	
uint8_t	periodicAdv↩ ListSize	

2.2.10 struct bleBondCreatedEvent_t

Bond Created Event.

Data Fields

uint8_t	nvmIndex	NVM index for the new created bond.
bleAddress↩ Type_t	addressType	Public or Random (static) address of the bond.
bleDevice↩ Address_t	address	Address of the bond.

2.2.11 struct gapAddrReadyEvent_t

Address Ready Event.

Data Fields

bleDevice↩ Address_t	aAddress	Generated device address.
uint8_t	advHandle	Advertising set handle if the generated device address will be used on an extended set. Reserved value 0xFF for other purposes↩ : legacy advertising or scanning and initiating address.

2.2.12 struct gapGenericEvent_t

Generic Event Structure = type + data.

Data Fields

gapGeneric↩ EventType_t	eventType	Event type.
union gapGeneric↩ Event_t	eventData	Event data, selected according to event type.

2.2.13 union gapGenericEvent_t.eventData

Data Fields

gapInternal↩ Error_t	internalError	Data for the gInternalError_c event. The error that has occurred and the command that triggered it.
uint8_t	whiteListSize	Data for the gWhiteListSizeReady_c event. The size of the White List.

bleDevice←Address_t	aAddress	Data for the gPublicAddressRead_c event. Contains the requested device address.
gapAddr←ReadyEvent_t	addrReady	Data for the gRandomAddressReady_c event. Contains the generated device address and advertising handle if applicable (0xFF otherwise).
uint8_t	advHandle	Data for the gRandomAddressSet_c event. Contains the handle of the configured advertising set or 0xFF for legacy advertising.
bleResult_t	setupFailError	Data for the gAdvertisingSetupFailed_c event. The error that occurred during the advertising setup.
int8_t	advTxPower←Level_dBm	Data for the gExtAdvertisingParametersSetupComplete_c and g←AdvTxPowerLevelRead_c events. Value in dBm.
bool_t	verified	Data for the gPrivateResolvableAddressVerified_c event. TRUE if the PRA was resolved with the given IRK.
gapLeScOob←Data_t	localOobData	Data for the gLeScLocalOobData_c event. Contains local OOB data for LESC Pairing.
bool_t	newHost←PrivacyState	Data for the gHostPrivacyStateChanged_c event. TRUE if enabled, FALSE if disabled.
bool_t	new←Controller←PrivacyState	Data for the gControllerPrivacyStateChanged_c event. TRUE if enabled, FALSE if disabled.
gapController←TestEvent_t	testEvent	Data for the gControllerTestEvent_c event. Contains test event type and received packets.
bleResult_t	txPowerLevel←SetStatus	Data for the gTxPowerLevelSetComplete_c event. Status of the set request.
gapPhyEvent←_t	phyEvent	Data for the gLePhyEvent_c event. Contains Tx and Rx Phy for a connection.
deviceId_t	deviceId	Data for the gTxEntryAvailable_c event.
gapInit←Complete_t	initComplete←Data	Data for the gInitializationComplete_c event. Contains the supported features, number of advertising sets and the size of the periodic advertiser list
ble←Notification←Event_t	notifEvent	Data for the gControllerNotificationEvent_c event. Contains status and adv/scan/conn event data.
bleBond←CreatedEvent←_t	bondCreated←Event	Data for the gBondCreatedEvent_c event. Contains the NVM index and the address of the bond.

2.2.14 struct bleBondIdentityHeaderBlob_t

Data Fields

uint32_t	raw[(gBleBondIdentityHeaderSize_c+3U)/sizeof(uint32_t)]	

2.2.15 struct bleBondDataDynamicBlob_t

Data Fields

uint32_t	raw[(gBleBondDataDynamicSize_c+3U)/sizeof(uint32_t)]	

2.2.16 struct bleBondDataStaticBlob_t

Data Fields

uint32_t	raw[(gBleBondDataStaticSize_c+3U)/sizeof(uint32_t)]	

2.2.17 struct bleBondDataDeviceInfoBlob_t

Data Fields

uint32_t	raw[(gBleBondDataDeviceInfoSize_c+3U)/sizeof(uint32_t)]	

2.2.18 struct bleBondDataDescriptorBlob_t

Data Fields

uint32_t	raw[(gBleBondDataDescriptorSize_c+3U)/sizeof(uint32_t)]	

2.2.19 struct bleBondDataBlob_t

Data Fields

bleBondIdentityHeaderBlob_t	bondHeader	
bleBondDataDynamicBlob_t	bondDataBlobDynamic	
bleBondDataStaticBlob_t	bondDataBlobStatic	
bleBondDataDescriptorBlob_t	bondDataDescriptors[gcGapMaximumSavedCccds_c]	
bleBondDataDeviceInfoBlob_t	bondDataBlobDeviceInfo	

2.3 Macro Definition Documentation

2.3.1 #define gcConnectionIntervalMin_c

Boundary values for the Connection Parameters (Standard GAP).

2.3.2 #define gcConnectionIntervalMinDefault_c

Default values for the Connection Parameters (Preferred).

connIntervalmin = Conn_Interval_Min $*$ 1.25 ms

Value of 0xFFFF indicates no specific minimum.

2.3.3 #define gcConnectionIntervalMaxDefault_c

connIntervalmax = Conn_Interval_Max ∗ 1.25 ms

Value of 0xFFFF indicates no specific maximum.

2.3.4 #define gcConnectionSupervisionTimeoutDefault_c

Time = N ∗ 10 ms.

2.3.5 #define gcConnectionEventMinDefault_c

Time = N ∗ 0.625 ms.

2.3.6 #define gcConnectionEventMaxDefault_c

Time = N ∗ 0.625 ms.

2.3.7 #define STATIC

When unit testing is performed, access from unit test module to static functions/variables within the tested module is not possible and therefore the static storage class identifier shall be removed.

2.3.8 #define gBleAddrTypePublic_c

Bluetooth Device Address Types.

Public Device Address - fixed into the Controller by the manufacturer.

2.3.9 #define gBleAddrTypeRandom_c

Random Device Address - set by the Host into the Controller for privacy reasons.

2.3.10 #define Ble_IsPrivateResolvableDeviceAddress(*bleAddress*)

PRA condition: check the 6th byte - MSB should be 0; 2nd MSB should be 1.

2.3.11 #define Ble_IsPrivateNonresolvableDeviceAddress(*bleAddress*)

PNRA condition: check the 6th byte - MSB should be 0; 2nd MSB should be 0.

2.3.12 #define Ble_IsRandomStaticDeviceAddress(*bleAddress*)

RSA condition: check the 6th byte - MSB should be 1; 2nd MSB should be 1.

2.3.13 #define Ble_DeviceAddressesMatch(*bleAddress1,* *bleAddress2*)

A macro used to compare two device addresses.

2.3.14 #define Ble_CopyDeviceAddress(*destinationAddress,* *sourceAddress*)

A macro used to copy device addresses.

2.3.15 #define gBleUuidType16_c

16-bit standard UUID

2.3.16 #define gBleUuidType128_c

128-bit long/custom UUID

2.3.17 #define gBleUuidType32_c

32-bit UUID - not available as ATT UUID format

2.3.18 #define gLePhy1MFlag_c

Host prefers to use LE 1M Tx/Rx Phy, possibly among others.

2.3.19 #define gLePhy2MFlag_c

Host prefers to use LE 2M Tx/Rx Phy, possibly among others.

2.3.20 #define gLePhyCodedFlag_c

Host prefers to use LE Coded Tx/Rx Phy, possibly among others.

2.3.21 #define gUseDeviceAddress_c

Initiator filter policy values.

Initiates a connection with a specific device identified by its address.

2.3.22 #define gUseWhiteList_c

Initiates connections with all the devices in the White List at the same time.

2.3.23 #define gScanAll_c

Scanning filter policy values.

Scans all advertising packets.

2.3.24 #define gScanWithWhiteList_c

Scans advertising packets using the White List.

2.3.25 #define gNetworkPrivacy_c

Privacy mode values.

Use Network Privacy Mode for the peer device (default)

2.3.26 #define gDevicePrivacy_c

Use Device Privacy Mode for the peer device.

2.3.27 #define gBleSig_PrimaryService_d

Bluetooth SIG UUID constants for GATT declarations.

Primary Service declaration UUID

2.3.28 #define gBleSig_SecondaryService_d

Secondary Service declaration UUID.

2.3.29 #define gBleSig_Include_d

Include declaration UUID.

2.3.30 #define gBleSig_Characteristic_d

Characteristic declaration UUID.

2.3.31 #define gBleSig_CCCD_d

Client Characteristic Configuration Descriptor declaration UUID.

2.3.32 #define gBleSig_SCCD_d

Server Characteristic Configuration Descriptor declaration UUID.

2.3.33 #define gBleSig_CharPresFormatDescriptor_d

Characteristic Presentation Format declaration UUID.

2.3.34 #define gBleSig_ValidRangeDescriptor_d

Valid Range Descriptor declaration UUID.

2.3.35 #define gBleSig_GenericAccessProfile_d

GAP Service UUID.

2.3.36 #define gBleSig_GenericAttributeProfile_d

GATT Service UUID.

2.3.37 #define gBleSig_ImmediateAlertService_d

Immediate Alert Service UUID.

2.3.38 #define gBleSig_LinkLossService_d

Link Loss Service UUID.

2.3.39 #define gBleSig_TxPowerService_d

Tx Power Service UUID.

2.3.40 #define gBleSig_CurrentTimeService_d

Current Time Service UUID.

2.3.41 #define gBleSig_ReferenceTimeUpdateService_d

Reference Time Update Service UUID.

2.3.42 #define gBleSig_NextDSTChangeService_d

Next DST Change Service UUID.

2.3.43 #define gBleSig_GlucoseService_d

Glucose Service UUID.

2.3.44 #define gBleSig_HealthThermometerService_d

Health Thermometer Service UUID.

2.3.45 #define gBleSig_DeviceInformationService_d

Device Information Service UUID.

2.3.46 #define gBleSig_HeartRateService_d

Heart Rate Service UUID.

2.3.47 #define gBleSig_PhoneAlertStatusService_d

Phone Alert Status Service UUID.

2.3.48 #define gBleSig_BatteryService_d

Battery Service UUID.

2.3.49 #define gBleSig_BloodPressureService_d

Blood Pressure Service UUID.

2.3.50 #define gBleSig_AlertNotificationService_d

Alert Notification Service UUID.

2.3.51 #define gBleSig_HidService_d

HID Service UUID.

2.3.52 #define gBleSig_RunningSpeedAndCadenceService_d

Running Speed And Cadence Service UUID.

2.3.53 #define gBleSig_CyclingSpeedAndCadenceService_d

Cycling Speed And Cadence Service UUID.

2.3.54 #define gBleSig_CyclingPowerService_d

Cycling Power Service UUID.

2.3.55 #define gBleSig_LocationAndNavigationService_d

Location And Navigation Service UUID.

2.3.56 #define gBleSig_IpsService_d

Internet Protocol Support Service UUID.

2.3.57 #define gBleSig_PulseOximeterService_d

Pulse Oximeter Service UUID.

2.3.58 #define gBleSig_HTTPProxyService_d

HTTP Proxy Service UUID.

2.3.59 #define gBleSig_WPTService_d

Wireless Power Transfer Service UUID.

2.3.60 #define gBleSig_BtpService_d

BTP Service UUID.

2.3.61 #define gBleSig_GapDeviceName_d

GAP Device Name Characteristic UUID.

2.3.62 #define gBleSig_GapAppearance_d

GAP Appearance Characteristic UUID.

2.3.63 #define gBleSig_GapPpcp_d

GAP Peripheral Preferred Connection Parameters Characteristic UUID.

2.3.64 #define gBleSig_GattServiceChanged_d

GATT Service Changed Characteristic UUID.

2.3.65 #define gBleSig_AlertLevel_d

Alert Level Characteristic UUID.

2.3.66 #define gBleSig_TxPower_d

TX Power Characteristic UUID.

2.3.67 #define gBleSig_LocalTimeInformation_d

Local Time Information Characteristic UUID.

2.3.68 #define gBleSig_TimeWithDST_d

Time With DST Characteristic UUID.

2.3.69 #define gBleSig_ReferenceTimeInformation_d

Reference Time Information Characteristic UUID.

2.3.70 #define gBleSig_TimeUpdateControlPoint_d

Time Update Control Point Characteristic UUID.

2.3.71 #define gBleSig_TimeUpdateState_d

Time Update State Characteristic UUID.

2.3.72 #define gBleSig_GlucoseMeasurement_d

Glucose Measurement Characteristic UUID.

2.3.73 #define gBleSig_BatteryLevel_d

Battery Level Characteristic UUID.

2.3.74 #define gBleSig_TemperatureMeasurement_d

Temperature Measurement Characteristic UUID.

2.3.75 #define gBleSig_TemperatureType_d

Temperature Type Characteristic UUID.

2.3.76 #define gBleSig_IntermediateTemperature_d

Intermediate Temperature Characteristic UUID.

2.3.77 #define gBleSig_MeasurementInterval_d

Measurement Interval Characteristic UUID.

2.3.78 #define gBleSig_SystemId_d

System ID Characteristic UUID.

2.3.79 #define gBleSig_ModelNumberString_d

Model Number String Characteristic UUID.

2.3.80 #define gBleSig_SerialNumberString_d

Serial Number String Characteristic UUID.

2.3.81 #define gBleSig_FirmwareRevisionString_d

Firmware Revision String Characteristic UUID.

2.3.82 #define gBleSig_HardwareRevisionString_d

Hardware Revision String Characteristic UUID.

2.3.83 #define gBleSig_SoftwareRevisionString_d

Software Revision String Characteristic UUID.

2.3.84 #define gBleSig_ManufacturerNameString_d

Manufacturer Name String Characteristic UUID.

2.3.85 #define gBleSig_IeeeRcdl_d

IEEE 11073-20601 Regulatory Certification Data List Characteristic UUID.

2.3.86 #define gBleSig_CurrentTime_d

Current Time Characteristic UUID.

2.3.87 #define gBleSig_BootKeyboardInputReport_d

Boot Keyboard Input Report UUID.

2.3.88 #define gBleSig_BootKeyboardOutputReport_d

Boot Keyboard output Report UUID.

2.3.89 #define gBleSig_BootMouseInputReport_d

Boot Mouse Input Report UUID.

2.3.90 #define gBleSig_GlucoseMeasurementContext_d

Glucose Measurement Context Characteristic UUID.

2.3.91 #define gBleSig_BpMeasurement_d

Blood Pressure Measurement UUID.

2.3.92 #define gBleSig_IntermediateCuffPressure_d

Intermediate Cuff Pressure UUID.

2.3.93 #define gBleSig_HrMeasurement_d

Heart Rate Measurement UUID.

2.3.94 #define gBleSig_BodySensorLocation_d

Body Sensor Location UUID.

2.3.95 #define gBleSig_HrControlPoint_d

Heart Rate Control Point UUID.

2.3.96 #define gBleSig_AlertStatus_d

Alert Status UUID.

2.3.97 #define gBleSig_RingerControlPoint_d

Ringer Control Point UUID.

2.3.98 #define gBleSig_RingerSetting_d

Ringer Setting UUID.

2.3.99 #define gBleSig_AlertNotifControlPoint_d

Alert Notif Control Point UUID.

2.3.100 #define gBleSig_UnreadAlertStatus_d

Unread Alert Status UUID.

2.3.101 #define gBleSig_NewAlert_d

New Alert UUID.

2.3.102 #define gBleSig_SupportedNewAlertCategory_d

Supported New Alert Category UUID.

2.3.103 #define gBleSig_SupportedUnreadAlertCategory_d

Supported Unread Alert Category UUID.

2.3.104 #define gBleSig_BloodPressureFeature_d

Blood Pressure Feature UUID.

2.3.105 #define gBleSig_HidInformation_d

HID Information UUID.

2.3.106 #define gBleSig_HidCtrlPoint_d

HID Control Point UUID.

2.3.107 #define gBleSig_Report_d

Report UUID.

2.3.108 #define gBleSig_ProtocolMode_d

Protocol Mode UUID.

2.3.109 #define gBleSig_ScanIntervalWindow_d

Scan Interval Window UUID.

2.3.110 #define gBleSig_PnpId_d

PnP Id UUID.

2.3.111 #define gBleSig_GlucoseFeature_d

Glucose Feature Characteristic UUID.

2.3.112 #define gBleSig_RaCtrlPoint_d

Record Access Ctrl Point Characteristic UUID.

2.3.113 #define gBleSig_RscMeasurement_d

RSC Measurement UUID.

2.3.114 #define gBleSig_RscFeature_d

RSC Feature UUID.

2.3.115 #define gBleSig_ScControlPoint_d

SC Control Point UUID.

2.3.116 #define gBleSig_CscMeasurement_d

CSC Measurement Characteristic UUID.

2.3.117 #define gBleSig_CscFeature_d

CSC Feature Characteristic UUID.

2.3.118 #define gBleSig_SensorLocation_d

Sensor Location Characteristic UUID.

2.3.119 #define gBleSig_PlxSCMeasurement_d

PLX Spot-Check Measurement Characteristic UUID.

2.3.120 #define gBleSig_PlxContMeasurement_d

PLX Continuous Measurement Characteristic UUID.

2.3.121 #define gBleSig_PulseOximeterFeature_d

PLX Feature Characteristic UUID.

2.3.122 #define gBleSig_CpMeasurement_d

CP Measurement Characteristic UUID.

2.3.123 #define gBleSig_CpVector_d

CP Measurement Vector UUID.

2.3.124 #define gBleSig_CpFeature_d

CP Feature CharacteristicUUID.

2.3.125 #define gBleSig_CpControlPoint_d

CP Control Point UUID.

2.3.126 #define gBleSig_LocationAndSpeed_d

Location and Speed Characteristic UUID.

2.3.127 #define gBleSig_Navigation_d

Navigation Characteristic UUID.

2.3.128 #define gBleSig_PositionQuality_d

Position Quality Characteristic UUID.

2.3.129 #define gBleSig_LnFeature_d

LN Feature Characteristic UUID.

2.3.130 #define gBleSig_LnControlPoint_d

LN Control Point Characteristic UUID.

2.3.131 #define gBleSig_Temperature_d

Temperature Characteristic UUID.

2.3.132 #define gBleSig_CentralAddressResolution_d

Central Address Resolution Characteristic UUID.

2.3.133 #define gBleSig_URI_d

URI Characteristic UUID.

2.3.134 #define gBleSig_HTTP_Headers_d

HTTP Headers Characteristic UUID.

2.3.135 #define gBleSig_HTTP_StatusCode_d

HTTP Status Code Characteristic UUID.

2.3.136 #define gBleSig_HTTP_EntityBody_d

HTTP Entity Body Characteristic UUID.

2.3.137 #define gBleSig_HTTP_ControlPoint_d

HTTP Control Point Characteristic UUID.

2.3.138 #define gBleSig_HTTPS_Security_d

HTTPS Security Characteristic UUID.

2.3.139 #define gBleSig_ResolvablePrivateAddressOnly_d

Resolvable Private Address Only Characteristic UUID.

2.3.140 #define gBleSig_MeshProvisioningService_d

BLE Mesh Provisioning Service UUID.

2.3.141 #define gBleSig_MeshProxyService_d

BLE Mesh Proxy Service UUID.

2.3.142 #define gBleSig_MeshProvDataIn_d

BLE Mesh Prov Data In Char UUID.

2.3.143 #define gBleSig_MeshProvDataOut_d

BLE Mesh Prov Data Out Char UUID.

2.3.144 #define gBleSig_MeshProxyDataIn_d

BLE Mesh Proxy Data In Char UUID.

2.3.145 #define gBleSig_MeshProxyDataOut_d

BLE Mesh Proxy Data Out Char UUID.

2.3.146 #define gBleSig_CAR_NotSupported_d

Central Address Resolution Characteristic Values.

2.3.147 #define gBleSig_RPAO_Used_d

Resolvable Private Address Only Characteristic Values.

2.3.148 #define BleSig_IsGroupingAttributeUuid16(*uuid16*)

Macro that returns whether or not an input 16-bit UUID is a grouping type.

2.3.149 #define BleSig_IsServiceDeclarationUuid16(*uuid16*)

Macro that returns whether or not an input 16-bit UUID is a Service declaration.

2.3.150 #define Uuid16(*uuid*)

Macro that declares a 16 bit UUID in a bleUuid_t union.

2.3.151 #define Uuid32(*uuid*)

Macro that declares a 32 bit UUID in a bleUuid_t union.

2.3.152 #define PACKED_STRUCT

Type qualifier - does not affect local variables of integral type.

2.3.153 #define global

Type qualifier - does not affect local variables of integral type.

Type qualifier - does not affect local variables of integral type

Storage class modifier - alignment of a variable. It does not affect the type of the function

Marks that this variable is in the interface.

2.3.154 #define __noreturn

Marks a function that never returns.

2.3.155 #define Utils_ExtractTwoByteValue(*buf*)

Returns a uint16_t from a buffer, little-endian.

2.3.156 #define Utils_ExtractThreeByteValue(*buf*)

Returns a 3-byte value from a buffer, little-endian.

2.3.157 #define Utils_ExtractFourByteValue(*buf*)

Returns a uint32_t from a buffer, little-endian.

2.3.158 #define Utils_BeExtractTwoByteValue(*buf*)

Returns a uint16_t from a buffer, big-endian.

2.3.159 #define Utils_BeExtractThreeByteValue(*buf*)

Returns a 3-byte value from a buffer, big-endian.

2.3.160 #define Utils_BeExtractFourByteValue(*buf*)

Returns a uint32_t from a buffer, big-endian.

2.3.161 #define Utils_PackTwoByteValue(*value,* *buf*)

Writes a uint16_t into a buffer, little-endian.

2.3.162 #define Utils_PackThreeByteValue(*value*, *buf*)

Writes a 3-byte value into a buffer, little-endian.

2.3.163 #define Utils_PackFourByteValue(*value*, *buf*)

Writes a uint32_t into a buffer, little-endian.

2.3.164 #define Utils_BePackTwoByteValue(*value*, *buf*)

Writes a uint16_t into a buffer, big-endian.

2.3.165 #define Utils_BePackThreeByteValue(*value*, *buf*)

Writes a 3-byte value into a buffer, big-endian.

2.3.166 #define Utils_BePackFourByteValue(*value*, *buf*)

Writes a uint32_t into a buffer, big-endian.

2.3.167 #define Utils_Copy8(*ptr*, *val8*)

Writes a uint8_t into a buffer, little-endian, and increments the pointer.

2.3.168 #define Utils_Copy16(*ptr*, *val16*)

Writes a uint16_t into a buffer, little-endian, and increments the pointer.

2.3.169 #define Utils_Copy32(*ptr*, *val32*)

Writes a uint32_t into a buffer, little-endian, and increments the pointer.

2.3.170 #define Utils_Copy64(*ptr*, *val64*)

Writes a uint64_t into a buffer, little-endian, and increments the pointer.

2.3.171 #define Utils_RevertByteArray(*array, size*)

Reverts the order of bytes in an array - useful for changing the endianness.

2.4 Typedef Documentation

2.4.1 typedef uint8_t deviceId_t

Unique identifier type for a connected device.

2.4.2 typedef uint8_t bleAddressType_t

Bluetooth Device Address Type - Size: 1 Octet, Range: [gBleAddrTypePublic_c:gBleAddrTypeRandom↩
_c].

2.4.3 typedef uint8_t bleDeviceAddress_t[gcBleDeviceAddressSize_c]

Bluetooth Device Address - array of 6 bytes.

2.4.4 typedef uint8_t bleUuidType_t

Bluetooth UUID type - values chosen to correspond with the ATT UUID format.

2.4.5 typedef uint16_t bleAdvReportEventProperties_t

Advertising Event properties.

2.4.6 typedef uint16_t bleAdvRequestProperties_t

Advertising Request properties.

2.4.7 typedef uint8_t bleScanningFilterPolicy_t

Scanning filter policy enumeration - Size: 1 Octet, Range: [gScanAll_c:gScanWithWhiteList_c].

2.4.8 typedef uint8_t bleInitiatorFilterPolicy_t

Initiator filter policy enumeration - Size: 1 Octet, Range: [gUseDeviceAddress_c:gUseWhiteList_c].

2.4.9 typedef uint8_t blePrivacyMode_t

Privacy Mode enumeration - Size: 1 Octet, Range: [gNetworkPrivacy_c:gDevicePrivacy_c].

2.4.10 typedef uint8_t bleChannelMap_t[gcBleChannelMapSize_c]

Bluetooth Channel map - array of 5 bytes.

2.4.11 typedef uint8_t gapLePhyFlags_t

Le Tx/Rx Phys Preferences flags.

2.4.12 typedef uint8_t gapLePhyMode_t

Le Tx/Rx Phys.

2.4.13 typedef uint16_t bleNotificationEventType_t

Controller Enhanced Notification Event Type.

2.4.14 typedef void(∗ gapGenericCallback_t) (gapGenericEvent_t ∗pGenericEvent)

Generic Callback prototype.

2.4.15 typedef bleResult_t(∗ hciHostToControllerInterface_t) (hciPacketType_t packetType, void ∗pPacket, uint16_t packetSize)

Host-to-Controller API prototype.

2.5 Enumeration Type Documentation

2.5.1 enum bleResult_t

BLE result type - the return value of BLE API functions.

Enumerator

gBleStatusBase_c	General status base.
gBleSuccess_c	Function executed successfully.

gBleInvalidParameter_c Parameter has an invalid value or is outside the accepted range.
gBleOverflow_c An internal limit is reached.
gBleUnavailable_c A requested parameter is not available.
gBleFeatureNotSupported_c The requested feature is not supported by this stack version.
gBleOutOfMemory_c An internal memory allocation failed.
gBleAlreadyInitialized_c Ble_HostInitialize function is incorrectly called a second time.
gBleOsError_c An error occurred at the OS level.
gBleUnexpectedError_c A "should never get here"-type error occurred.
gBleInvalidState_c The requested API cannot be called in the current state.
gBleTimerError_c Timer allocation failed.
gSmCommandNotSupported_c The Security Manager (SM) does not have the required features or version to support this command.
gSmUnexpectedCommand_c This command is not or cannot be handled in the current context of the SM.
gSmInvalidCommandCode_c The provided SM command code is invalid.
gSmInvalidCommandLength_c The provided command length is not valid for the SM command code.
gSmInvalidCommandParameter_c One of the parameters of the SM command is not valid.
gSmInvalidDeviceId_c The provided Device ID is invalid.
gSmInvalidInternalOperation_c There is a problem with the internal state of the SM. This should not happen during normal operation. A memory corruption or invalid operation may have occurred.
gSmInvalidConnectionHandle_c The target device does not have a valid connection handle. It might be disconnected.
gSmInproperKeyDistributionField_c The Responder upper layer has set to "1" one or more flags in the Initiator or Responder Key Distribution Fields from the Pairing Request which were set to "0" by the peer device.
gSmUnexpectedKeyType_c The Responder upper layer has set a key type field in the Passkey Request Reply command, which is different than the field negotiated with the peer device.
gSmUnexpectedPairingTerminationReason_c The upper layer tried to cancel the pairing procedure with an unexpected pairing failure reason for the current phase of the pairing procedure.
gSmUnexpectedKeyset_c The Responder upper layer is trying to distribute keys which were not requested during the pairing procedure or the peer device has sent a Key Distribution packet which was not expected.
gSmSmpTimeoutOccurred_c An SMP timeout has occurred for the peer device. No more operations are accepted until a new physical link is established.
gSmUnknownSmpPacketType_c An SMP packet with an unknown (or invalid) type has been received.
gSmInvalidSmpPacketLength_c An SMP packet with an invalid length for the SMP packet type has been received.
gSmInvalidSmpPacketParameter_c An SMP packet with an invalid parameter has been received.
gSmReceivedUnexpectedSmpPacket_c An unexpected SMP packet was received.
gSmReceivedSmpPacketFromUnknownDevice_c An SMP packet is received but the source Device ID cannot be identified.
gSmReceivedUnexpectedHciEvent_c An HCI event has been received which cannot be handled by

the SM or cannot be handled in the current context.

gSmReceivedHciEventFromUnknownDevice_c An HCI event is received but the source Device ID cannot be identified.

gSmInvalidHciEventParameter_c An HCI Event is received with an invalid parameter.

gSmLlConnectionEncryptionInProgress_c A Link Layer Connection encryption was requested by the upper layer or attempted internally by the SM, but it could no be completed because an encryption was already in progress. This situation could lead to an SMP Pairing Failure when the SM cannot encrypt the link with the STK. An unspecified pairing failure reason is used in this instance.

gSmLlConnectionEncryptionFailure_c The Link Layer connection encryption procedure has failed.

gSmInsufficientResources_c The SM could not allocate resources to perform operations (memory or timers).

gSmOobDataAddressMismatch_c The address of the peer contained in the remote OOB data sent to the stack does not match the address used by the remote device for the connection/pairing procedure.

gSmSmpPacketReceivedAfterTimeoutOccurred_c A SMP packet has been received from a peer device for which a pairing procedure has timed out. No further operations are permitted until a new connection is established.

gSmReceivedTimerEventForUnknownDevice_c An Timer event is received but the source Device ID cannot be identified.

gSmUnattainableLocalDeviceSecRequirements_c The provided pairing parameters cannot lead to a Pairing Procedure which satisfies the minimum security properties for the local device.

gSmUnattainableLocalDeviceMinKeySize_c The provided pairing parameters cannot lead to a Pairing Procedure which satisfies the minimum encryption key size for the local device.

gSmUnattainableSlaveSecReqRequirements_c The provided pairing parameters cannot lead to a Pairing Procedure which satisfies the minimum security properties requested by the local device via a SMP Slave Security Request.

gSmTbResolvableAddressDoesNotMatchIrk_c The provided Resolvable Private Address and IRK do not match.

gSmTbInvalidDataSignature_c The provided data signature does not match the computed data signature.

gAttStatusBase_c ATT status base.

gAttSuccess_c Alias.

gGattStatusBase_c GATT status base.

gGattSuccess_c Alias.

gGattAnotherProcedureInProgress_c Trying to start a GATT procedure while one is already in progress.

gGattLongAttributePacketsCorrupted_c Writing a Long Characteristic failed because Prepare Write Request packets were corrupted.

gGattMultipleAttributesOverflow_c Too many Characteristics are given for a Read Multiple Characteristic procedure.

gGattUnexpectedReadMultipleResponseLength_c Read Multiple Characteristic procedure failed because unexpectedly long data was read.

gGattInvalidValueLength_c An invalid value length was supplied to a Characteristic Read/Write operation.

gGattServerTimeout_c No response was received from the Server.

gGattIndicationAlreadyInProgress_c A Server Indication is already waiting for Client Confirmation.

gGattClientConfirmationTimeout_c No Confirmation was received from the Client after a Server Indication.

gGattInvalidPduReceived_c An invalid PDU length was received.

gGapStatusBase_c GAP status base.

gGapSuccess_c Alias.

gGapAdvDataTooLong_c Trying to set too many bytes in the advertising payload.

gGapScanRspDataTooLong_c Trying to set too many bytes in the scan response payload.

gGapDeviceNotBonded_c Trying to execute an API that is only available for bonded devices.

gGapAnotherProcedureInProgress_c Trying to start a GAP procedure while one is already in progress.

gDevDbStatusBase_c DeviceDatabase status base.

gDevDbSuccess_c Alias.

gDevDbCccdLimitReached_c CCCD value cannot be saved because Server's CCCD list is full for the current client.

gDevDbCccdNotFound_c CCCD with the given handle is not found in the Server's list for the current client.

gGattDbStatusBase_c GATT Database status base.

gGattDbSuccess_c Alias.

gGattDbInvalidHandle_c An invalid handle was passed as parameter.

gGattDbCharacteristicNotFound_c Characteristic was not found.

gGattDbCccdNotFound_c CCCD was not found.

gGattDbServiceNotFound_c Service Declaration was not found.

gGattDbDescriptorNotFound_c Characteristic Descriptor was not found.

2.5.2 enum bleAdvertisingType_t

Advertising Type.

Enumerator

gAdvConnectableUndirected_c Answers to both connect and scan requests.

gAdvDirectedHighDutyCycle_c Answers only to connect requests; smaller advertising interval for quicker connection.

gAdvScannable_c Answers only to scan requests.

gAdvNonConnectable_c Does not answer to connect nor scan requests.

gAdvDirectedLowDutyCycle_c Answers only to connect requests; larger advertising interval.

2.5.3 enum bleAdvReportEventProperties_tag

Enumerator

gAdvEventConnectable_c	Connectable Advertisement.
gAdvEventScannable_c	Scannable Advertisement.
gAdvEventDirected_c	Directed Advertisement.
gAdvEventScanResponse_c	Scan Response.
gAdvEventLegacy_c	Legacy Advertisement PDU.
gAdvEventAnonymous_c	Anonymous Advertisement.

2.5.4 enum bleAdvRequestProperties_tag

Enumerator

gAdvReqConnectable_c	Connectable Advertising.
gAdvReqScannable_c	Scannable Advertising.
gAdvReqDirected_c	Directed Advertising.
gAdvReqHighDutyCycle_c	High Duty Cycle.
gAdvReqLegacy_c	Legacy Advertising PDU.
gAdvReqAnonymous_c	Anonymous Advertising.
gAdvIncludeTxPower_c	Set this option to include the Tx power in advertising packet.

2.5.5 enum bleAdvertisingFilterPolicy_t

Enumerator

gBleAdvFilterAllowScanFromAnyAllowConnFromAny_c	White List is ignored.
gBleAdvFilterAllowScanFromWLAllowConnFromAny_c	White List is used only for Scan Requests.
gBleAdvFilterAllowScanFromAnyAllowConnFromWL_c	White List is used only for Connection Requests.
gBleAdvFilterAllowScanFromWLAllowConnFromWL_c	White List is used for both Scan and Connection Requests.

2.5.6 enum bleLlConnectionRole_t

Enumerator

gBleLlConnectionMaster_c	Link Layer Master Role.
gBleLlConnectionSlave_c	Link Layer Slave Role.

2.5.7 enum hciPacketType_t

Enumerator

gHciCommandPacket_c HCI Command.
gHciDataPacket_c L2CAP Data Packet.
gHciSynchronousDataPacket_c Not used in BLE.
gHciEventPacket_c HCI Event.

2.5.8 enum bleScanType_t

Scanning type enumeration.

Enumerator

gScanTypePassive_c Passive Scanning - advertising packets are immediately reported to the Host.
gScanTypeActive_c Active Scanning - the scanner sends scan requests to the advertiser and reports to the Host after the scan response is received.

2.5.9 enum bleTransmitPowerLevelType_t

Enumerator

gReadCurrentTxPowerLevel_c Current TX Power level.
gReadMaximumTxPowerLevel_c Maximum recorded TX Power level.

2.5.10 enum bleTransmitPowerChannelType_t

Enumerator

gTxPowerAdvChannel_c Advertising channel type when setting Tx Power.
gTxPowerConnChannel_c Connection channel type when setting Tx Power.

2.5.11 enum gapGenericEventType_t

Generic Event Type.

Enumerator

gInitializationComplete_c Initial setup started by Ble_HostInitialize is complete.

gInternalError_c An internal error occurred.

gAdvertisingSetupFailed_c Error during advertising setup.

gAdvertisingParametersSetupComplete_c Advertising parameters have been successfully set. Response to Gap_SetAdvertisingParameters.

gAdvertisingDataSetupComplete_c Advertising and/or scan response data has been successfully set. Response to Gap_SetAdvertisingData.

gWhiteListSizeRead_c Contains the White List size. Response to Gap_ReadWhiteListSize.

gDeviceAddedToWhiteList_c Device has been added to White List. Response to Gap_AddDevice↩ ToWhiteList.

gDeviceRemovedFromWhiteList_c Device has been removed from the White List. Response to Gap_RemoveDeviceFromWhiteList.

gWhiteListCleared_c White List has been cleared. Response to Gap_ClearWhiteList.

gRandomAddressReady_c A random device address has been created. Response to Gap_Create↩ RandomDeviceAddress.

gCreateConnectionCanceled_c Connection initiation was successfully cancelled. Response to Gap_CancelInitiatingConnection.

gPublicAddressRead_c Contains the public device address. Response to Gap_ReadPublicDevice↩ Address.

gAdvTxPowerLevelRead_c Contains the TX power on the advertising channel. Response to Gap_↩ ReadAdvertisingTxPowerLevel.

gPrivateResolvableAddressVerified_c Contains the result of PRA verification. Response to Gap_↩ VerifyPrivateResolvableAddress.

gRandomAddressSet_c Random address has been set into the Controller. Response to Gap_Set↩ RandomAddress.

gLeScPublicKeyRegenerated_c The private/public key pair used for LE Secure Connections pairing has been regenerated.

gLeScLocalOobData_c Local OOB data used for LE Secure Connections pairing.

gHostPrivacyStateChanged_c Host Privacy was enabled or disabled.

gControllerPrivacyStateChanged_c Controller Privacy was enabled or disabled.

gControllerTestEvent_c Controller Test was started or stopped.

gTxPowerLevelSetComplete_c Controller Tx Power Level set complete or invalid.

gLePhyEvent_c Phy Mode of a connection has been updated by the Controller.

gControllerNotificationEvent_c Controller Enhanced Notification received.

gBondCreatedEvent_c Bond Created Event signalling the stack created a bond after pairing or at app request.

gChannelMapSet_c Channel map set complete in the Controller.

gExtAdvertisingParametersSetupComplete_c Extended advertising parameters have been successfully set.

gExtAdvertisingDataSetupComplete_c Extended advertising data has been successfully set.

gExtAdvertisingSetRemoveComplete_c An advertising set has been removed from the Controller.

gPeriodicAdvParamSetupComplete_c Periodic advertising parameters have been successfully set.

gPeriodicAdvDataSetupComplete_c Periodic advertising data have been successfully set.

gPeriodicAdvListUpdateComplete_c Periodic advertiser list has been successfully updated.

gPeriodicAdvCreateSyncCancelled_c Periodic advertising create sync command was successfully cancelled.

gTxEntryAvailable_c This event is generated when a TX entry becomes available after they were all in use.

2.5.12 enum gapInternalErrorSource_t

Internal Error Source - the command that triggered the error.

2.5.13 enum gapControllerTestEventType_t

Controller Test Event Type.

2.5.14 enum gapLeAllPhyFlags_t

Le All Phys Preferences flags.

Enumerator

gLeTxPhyNoPreference_c	Host has no preference for Tx Phy.
gLeRxPhyNoPreference_c	Host has no preference for Rx Phy.

2.5.15 enum gapLePhyOptionsFlags_t

Le Phys Options Preferences flags.

Enumerator

gLeCodingNoPreference_c	Host has no preference on the LE Coded Phy.
gLeCodingS2_c	Host prefers to use S=2 on the LE Coded Phy.
gLeCodingS8_c	Host prefers to use S=8 on the LE Coded Phy.

2.5.16 enum gapLePhyMode_tag

Enumerator

gLePhy1M_c	Tx/Rx Phy on the connection is LE 1M.
gLePhy2M_c	Tx/Rx Phy on the connection is LE 2M.
gLePhyCoded_c	Tx/Rx Phy on the connection is LE Coded.

2.5.17 enum gapPhyEventType_t

Phy Event Type.

Enumerator

gPhySetDefaultComplete_c Gap_LeSetPhy default mode was successful.
gPhyRead_c Gap_LeReadPhy return values.
gPhyUpdateComplete_c Gap_LeSetPhy return values for a connection or an update occurred.

2.5.18 enum bleNotificationEventType_tag

Enumerator

gNotifEventNone_c No enhanced notification event enabled.
gNotifConnEventOver_c Connection event over.
gNotifConnRxPdu_c Connection Rx PDU.
gNotifAdvEventOver_c Advertising event over.
gNotifAdvTx_c Advertising ADV transmitted.
gNotifAdvScanReqRx_c Advertising SCAN REQ Rx.
gNotifAdvConnReqRx_c Advertising CONN REQ Rx.
gNotifScanEventOver_c Scanning event over.
gNotifScanAdvPktRx_c Scanning ADV PKT Rx.
gNotifScanRspRx_c Scanning SCAN RSP Rx.
gNotifScanReqTx_c Scanning SCAN REQ Tx.
gNotifConnCreated_c Connection created.

2.6 Function Documentation

2.6.1 bleResult_t Ble_HostInitialize (gapGenericCallback_t *genericCallback*, hciHostToControllerInterface_t *hostToControllerInterface*)

Performs master initialization of the BLE Host stack.

Parameters

`in`	*generic↩ Callback*	Callback used to propagate GAP generic events to the application.
`in`	*hostTo↩ Controller↩ Interface*	LE Controller uplink interface function pointer

Returns

gBleSuccess_c or error.

Remarks

Application must wait for the gInitializationComplete_c generic event.

2.6.2 bleResult_t Ble_HciRecv (hciPacketType_t *packetType,* void ∗ *pHciPacket,* uint16_t *packetSize*)

This is the BLE Host downlink interface function.

Parameters

in	*packetType*	The type of the packet sent by the LE Controller
in	*pHciPacket*	Pointer to the packet sent by the LE Controller
in	*packetSize*	Number of bytes sent by the LE Controller

Returns

gBleSuccess_c or gBleOutOfMemory_c

Remarks

This function must be registered as a callback by the LE Controller and called to send HCI packets (events and LE-U data) to the BLE Host.

2.6.3 void Host_TaskHandler (void ∗ *args*)

Contains the Host Task logic.

Remarks

This function must be called exclusively by the Host Task code from the application.

2.7 Variable Documentation

2.7.1 msgQueue_t gApp2Host_TaskQueue

App to Host message queue for the Host Task.

2.7.2 msgQueue_t gHci2Host_TaskQueue

HCI to Host message queue for the Host Task.

2.7.3 osaEventId_t gHost_TaskEvent

Event for the Host Task Queue.

Chapter 3
Generic Access Profile

3.1 Overview

Files

- file gap_interface.h
- file gap_types.h

Data Structures

- struct gapSmpKeys_t
- struct gapSecurityRequirements_t
- struct gapServiceSecurityRequirements_t
- struct gapDeviceSecurityRequirements_t
- struct gapHandleList_t
- struct gapConnectionSecurityInformation_t
- struct gapPairingParameters_t
- struct gapSlaveSecurityRequestParameters_t
- struct gapAdvertisingParameters_t
- struct gapExtAdvertisingParameters_t
- struct gapPeriodicAdvParameters_t
- struct gapScanningParameters_t
- struct gapPeriodicAdvSyncReq_t
- struct gapConnectionRequestParameters_t
- struct gapConnectionParameters_t
- struct gapAdStructure_t
- struct gapAdvertisingData_t
- struct gapExtScanNotification_t
- struct gapAdvertisingSetTerminated_t
- struct gapAdvertisingEvent_t
- union gapAdvertisingEvent_t.eventData
- struct gapScannedDevice_t
- struct gapExtScannedDevice_t
- struct gapPeriodicScannedDevice_t
- struct gapSyncEstbEventData_t
- struct gapSyncLostEventData_t
- struct gapScanningEvent_t
- union gapScanningEvent_t.eventData
- struct gapConnectedEvent_t
- struct gapKeyExchangeRequestEvent_t
- struct gapKeysReceivedEvent_t
- struct gapAuthenticationRejectedEvent_t
- struct gapPairingCompleteEvent_t
- union gapPairingCompleteEvent_t.pairingCompleteData
- struct gapLongTermKeyRequestEvent_t
- struct gapEncryptionChangedEvent_t
- struct gapDisconnectedEvent_t

- struct gapConnParamsUpdateReq_t
- struct gapConnParamsUpdateComplete_t
- struct gapConnLeDataLengthChanged_t
- struct gapConnectionEvent_t
- union gapConnectionEvent_t.eventData
- struct gapIdentityInformation_t
- struct gapAutoConnectParams_t

Macros

- #define Gap_AddSecurityModesAndLevels(modeLevelA, modeLevelB)
- #define Gap_CancelInitiatingConnection()
- #define Gap_ReadAdvertisingTxPowerLevel()
- #define Gap_ReadRssi(deviceId)
- #define Gap_ReadTxPowerLevelInConnection(deviceId)
- #define gCancelOngoingInitiatingConnection_d
- #define gMode_2_Mask_d
- #define getSecurityLevel(modeLevel)
- #define getSecurityMode(modeLevel)
- #define **isMode_2**(modeLevel)
- #define **isMode_1**(modeLevel)
- #define **isSameMode**(modeLevelA, modeLevelB)
- #define **addSameSecurityModes**(modeLevelA, modeLevelB)
- #define **addMode1AndMode2**(mode1, mode2)
- #define **addDifferentSecurityModes**(modeLevelA, modeLevelB)
- #define gDefaultEncryptionKeySize_d
- #define gMaxEncryptionKeySize_d
- #define gGapDefaultDeviceSecurity_d
- #define gGapDefaultSecurityRequirements_d
- #define gGapAdvertisingIntervalRangeMinimum_c
- #define gGapAdvertisingIntervalDefault_c
- #define gGapAdvertisingIntervalRangeMaximum_c
- #define gGapExtAdvertisingIntervalRangeMinimum_c
- #define gGapExtAdvertisingIntervalDefault_c
- #define gGapExtAdvertisingIntervalRangeMaximum_c
- #define gGapPeriodicAdvIntervalRangeMinimum_c
- #define gGapPeriodicAdvIntervalDefault_c
- #define gGapPeriodicAdvIntervalRangeMaximum_c
- #define gGapAdvertisingChannelMapDefault_c
- #define gGapDefaultAdvertisingParameters_d
- #define gGapDefaultExtAdvertisingParameters_d
- #define gGapDefaultPeriodicAdvParameters_d
- #define gGapScanIntervalMin_d
- #define gGapScanIntervalDefault_d
- #define gGapScanIntervalMax_d
- #define gGapScanWindowMin_d
- #define gGapScanWindowDefault_d
- #define gGapScanWindowMax_d
- #define gGapRssiMin_d
- #define gGapRssiMax_d
- #define gGapRssiNotAvailable_d
- #define gGapScanContinuously_d
- #define gGapScanPeriodicDisabled_d
- #define gGapDefaultScanningParameters_d
- #define gGapConnIntervalMin_d
- #define gGapConnIntervalMax_d

- #define gGapConnLatencyMin_d
- #define gGapConnLatencyMax_d
- #define gGapConnSuperTimeoutMin_d
- #define gGapConnSuperTimeoutMax_d
- #define gGapConnEventLengthMin_d
- #define gGapConnEventLengthMax_d
- #define gGapDefaultConnectionLatency_d
- #define gGapDefaultSupervisionTimeout_d
- #define gGapDefaultMinConnectionInterval_d
- #define gGapDefaultMaxConnectionInterval_d
- #define gGapDefaultConnectionRequestParameters_d
- #define gGapChSelAlgorithmNo2
- #define gBlePeriodicAdvOngoingSyncCancelHandle
- #define gGapInvalidSyncHandle
- #define gNone_c
- #define gLeLimitedDiscoverableMode_c
- #define gLeGeneralDiscoverableMode_c
- #define gBrEdrNotSupported_c
- #define gSimultaneousLeBrEdrCapableController_c
- #define gSimultaneousLeBrEdrCapableHost_c
- #define gNoKeys_c
- #define gLtk_c
- #define gIrk_c
- #define gCsrk_c
- #define gSecurityMode_1_c
- #define gSecurityMode_2_c
- #define gSecurityLevel_NoSecurity_c
- #define gSecurityLevel_NoMitmProtection_c
- #define gSecurityLevel_WithMitmProtection_c
- #define gSecurityLevel_LeSecureConnections_c
- #define gSecurityMode_1_Level_1_c
- #define gSecurityMode_1_Level_2_c
- #define gSecurityMode_1_Level_3_c
- #define gSecurityMode_1_Level_4_c
- #define gSecurityMode_2_Level_1_c
- #define gSecurityMode_2_Level_2_c
- #define gOobNotAvailable_c
- #define gIncompatibleIoCapabilities_c
- #define gPairingNotSupported_c
- #define gLowEncryptionKeySize_c
- #define gUnspecifiedReason_c
- #define gRepeatedAttempts_c
- #define gLinkEncryptionFailed_c
- #define gIoDisplayOnly_c
- #define gIoDisplayYesNo_c
- #define gIoKeyboardOnly_c
- #define gIoNone_c
- #define gIoKeyboardDisplay_c

Typedefs

- typedef uint8_t gapIoCapabilities_t
- typedef uint8_t gapSmpKeyFlags_t
- typedef uint8_t gapSecurityMode_t
- typedef uint8_t gapSecurityLevel_t
- typedef uint8_t gapSecurityModeAndLevel_t

- typedef uint8_t gapKeypressNotification_t
- typedef uint8_t gapAuthenticationRejectReason_t
- typedef uint8_t gapCreateSyncReqFilterPolicy_t
- typedef uint8_t gapAdTypeFlags_t
- typedef gapAdvertisingData_t gapScanResponseData_t
- typedef uint8_t gapControllerTestTxType_t
- typedef bleResult_t gapDisconnectionReason_t
- typedef void(∗ gapAdvertisingCallback_t) (gapAdvertisingEvent_t ∗pAdvertisingEvent)
- typedef void(∗ gapScanningCallback_t) (gapScanningEvent_t ∗pScanningEvent)
- typedef void(∗ gapConnectionCallback_t) (deviceId_t deviceId, gapConnectionEvent_t ∗p↩ConnectionEvent)

Enumerations

- enum gapRole_t {
  gGapCentral_c,
  gGapPeripheral_c,
  gGapObserver_c,
  gGapBroadcaster_c }
- enum gapKeypressNotification_tag {
  gKnPasskeyEntryStarted_c,
  gKnPasskeyDigitStarted_c,
  gKnPasskeyDigitErased_c,
  gKnPasskeyCleared_c,
  gKnPasskeyEntryCompleted_c }
- enum gapScanMode_t {
  gDefaultScan_c,
  gLimitedDiscovery_c,
  gGeneralDiscovery_c,
  gAutoConnect_c }
- enum gapAdvertisingChannelMapFlags_t {
  gAdvChanMapFlag37_c,
  gAdvChanMapFlag38_c,
  gAdvChanMapFlag39_c }
- enum gapAdvertisingFilterPolicy_t {
  gProcessAll_c,
  gProcessConnAllScanWL_c,
  gProcessScanAllConnWL_c,
  gProcessWhiteListOnly_c }
- enum gapFilterDuplicates_t {
  gGapDuplicateFilteringDisable_c,
  gGapDuplicateFilteringEnable_c,
  gGapDuplicateFilteringPeriodicEnable_c }
- enum gapCreateSyncReqFilterPolicy_tag {
  gUseCommandParameters_c,
  gUsePeriodicAdvList_c }
- enum gapAdType_t {

gAdFlags_c,
gAdIncomplete16bitServiceList_c,
gAdComplete16bitServiceList_c,
gAdIncomplete32bitServiceList_c,
gAdComplete32bitServiceList_c,
gAdIncomplete128bitServiceList_c,
gAdComplete128bitServiceList_c,
gAdShortenedLocalName_c,
gAdCompleteLocalName_c,
gAdTxPowerLevel_c,
gAdClassOfDevice_c,
gAdSimplePairingHashC192_c,
gAdSimplePairingRandomizerR192_c,
gAdSecurityManagerTkValue_c,
gAdSecurityManagerOobFlags_c,
gAdSlaveConnectionIntervalRange_c,
gAdServiceSolicitationList16bit_c,
gAdServiceSolicitationList32bit_c,
gAdServiceSolicitationList128bit_c,
gAdServiceData16bit_c,
gAdServiceData32bit_c,
gAdServiceData128bit_c,
gAdPublicTargetAddress_c,
gAdRandomTargetAddress_c,
gAdAppearance_c,
gAdAdvertisingInterval_c,
gAdLeDeviceAddress_c,
gAdLeRole_c,
gAdSimplePairingHashC256_c,
gAdSimplePairingRandomizerR256_c,
gAd3dInformationData_c,
gAdUniformResourceIdentifier_c,
gAdLeSupportedFeatures_c,
gAdChannelMapUpdateIndication_c,
gAdManufacturerSpecificData_c }

- enum gapRadioPowerLevelReadType_t {
  gTxPowerCurrentLevelInConnection_c,
  gTxPowerMaximumLevelInConnection_c,
  gTxPowerLevelForAdvertising_c,
  gRssi_c }
- enum gapControllerTestCmd_t {
  gControllerTestCmdStartRx_c,
  gControllerTestCmdStartTx_c,
  gControllerTestCmdEnd_c }
- enum gapControllerTestTxType_tag {

gControllerTestTxPrbs9_c,
gControllerTestTxF0_c,
gControllerTestTxAA_c,
gControllerTestTxPrbs15_c,
gControllerTestTxFF_c,
gControllerTestTx00_c,
gControllerTestTx0F_c,
gControllerTestTx55_c }
- enum gapAdvertisingEventType_t {
gAdvertisingStateChanged_c,
gAdvertisingCommandFailed_c,
gExtAdvertisingStateChanged_c,
gAdvertisingSetTerminated_c,
gExtScanNotification_c,
gPeriodicAdvertisingStateChanged_c }
- enum gapScanningEventType_t {
gScanStateChanged_c,
gScanCommandFailed_c,
gDeviceScanned_c,
gExtDeviceScanned_c,
gPeriodicDeviceScanned_c,
gPeriodicAdvSyncEstablished_c,
gPeriodicAdvSyncLost_c,
gPeriodicAdvSyncTerminated_c }
- enum gapConnectionEventType_t {

gConnEvtConnected_c,
gConnEvtPairingRequest_c,
gConnEvtSlaveSecurityRequest_c,
gConnEvtPairingResponse_c,
gConnEvtAuthenticationRejected_c,
gConnEvtPasskeyRequest_c,
gConnEvtOobRequest_c,
gConnEvtPasskeyDisplay_c,
gConnEvtKeyExchangeRequest_c,
gConnEvtKeysReceived_c,
gConnEvtLongTermKeyRequest_c,
gConnEvtEncryptionChanged_c,
gConnEvtPairingComplete_c,
gConnEvtDisconnected_c,
gConnEvtRssiRead_c,
gConnEvtTxPowerLevelRead_c,
gConnEvtPowerReadFailure_c,
gConnEvtParameterUpdateRequest_c,
gConnEvtParameterUpdateComplete_c,
gConnEvtLeDataLengthChanged_c,
gConnEvtLeScOobDataRequest_c,
gConnEvtLeScDisplayNumericValue_c,
gConnEvtLeScKeypressNotification_c,
gConnEvtChannelMapRead_c,
gConnEvtChannelMapReadFailure_c,
gConnEvtChanSelectionAlgorithm2_c,
gConnEvtPairingNoLtk_c }

- enum gapCarSupport_t {
  CAR_Unknown,
  CAR_Unavailable,
  CAR_Unsupported,
  CAR_Supported }
- enum **gapPeriodicAdvListOperation_t** {
  gAddDevice_c,
  gRemoveDevice_c,
  gRemoveAllDevices_c }
- enum gapAppearance_t {

gUnknown_c,
gGenericPhone_c,
gGenericComputer_c,
gGenericWatch_c,
gSportsWatch_c,
gGenericClock_c,
gGenericDisplay_c,
gGenericRemoteControl_c,
gGenericEyeglasses_c,
gGenericTag_c,
gGenericKeyring_c,
gGenericMediaPlayer_c,
gGenericBarcodeScanner_c,
gGenericThermometer_c,
gThermometerEar_c,
gGenericHeartrateSensor_c,
gHeartRateSensorHeartRateBelt_c,
gGenericBloodPressure_c,
gBloodPressureArm_c,
gBloodPressureWrist_c,
gHumanInterfaceDevice_c,
gKeyboard_c,
gMouse_c,
gJoystick_c,
gGamepad_c,
gDigitizerTablet_c,
gCardReader_c,
gDigitalPen_c,
gBarcodeScanner_c,
gGenericGlucoseMeter_c,
gGenericRunningWalkingSensor_c,
gRunningWalkingSensorInShoe_c,
gRunningWalkingSensorOnShoe_c,
gRunningWalkingSensorOnHip_c,
gGenericCycling_c,
gCyclingComputer_c,
gCyclingSpeedSensor_c,
gCyclingCadenceSensor_c,
gCyclingPowerSensor_c,
gCyclingSpeedandCadenceSensor_c,
gGenericPulseOximeter_c,
gFingertip_c,
gWristWorn_c,
gGenericWeightScale_c,
gGenericOutdoorSportsActivity_c,
gLocationDisplayDevice_c,
gLocationandNavigationDisplayDevice_c,
gLocationPod_c,

gLocationAndNavigationPod_c }

Functions

- bleResult_t Gap_RegisterDeviceSecurityRequirements (const gapDeviceSecurityRequirements_↩ t ∗pSecurity)
- bleResult_t Gap_SetAdvertisingParameters (const gapAdvertisingParameters_t ∗pAdvertising↩ Parameters)
- bleResult_t Gap_SetAdvertisingData (const gapAdvertisingData_t ∗pAdvertisingData, const gap↩ ScanResponseData_t ∗pScanResponseData)
- bleResult_t Gap_StartAdvertising (gapAdvertisingCallback_t advertisingCallback, gapConnection↩ Callback_t connectionCallback)
- bleResult_t Gap_StopAdvertising (void)
- bleResult_t Gap_Authorize (deviceId_t deviceId, uint16_t handle, gattDbAccessType_t access)
- bleResult_t Gap_SaveCccd (deviceId_t deviceId, uint16_t handle, gattCccdFlags_t cccd)
- bleResult_t Gap_CheckNotificationStatus (deviceId_t deviceId, uint16_t handle, bool_t ∗pOutIs↩ Active)
- bleResult_t Gap_CheckIndicationStatus (deviceId_t deviceId, uint16_t handle, bool_t ∗pOutIs↩ Active)
- bleResult_t Gap_GetBondedDevicesIdentityInformation (gapIdentityInformation_t ∗aOutIdentity↩ Addresses, uint8_t maxDevices, uint8_t ∗pOutActualCount)
- bleResult_t Gap_Pair (deviceId_t deviceId, const gapPairingParameters_t ∗pPairingParameters)
- bleResult_t Gap_SendSlaveSecurityRequest (deviceId_t deviceId, const gapPairingParameters_↩ t ∗pPairingParameters)
- bleResult_t Gap_EncryptLink (deviceId_t deviceId)
- bleResult_t Gap_AcceptPairingRequest (deviceId_t deviceId, const gapPairingParameters_t ∗p↩ PairingParameters)
- bleResult_t Gap_RejectPairing (deviceId_t deviceId, gapAuthenticationRejectReason_t reason)
- bleResult_t Gap_EnterPasskey (deviceId_t deviceId, uint32_t passkey)
- bleResult_t Gap_ProvideOob (deviceId_t deviceId, const uint8_t ∗aOob)
- bleResult_t Gap_RejectPasskeyRequest (deviceId_t deviceId)
- bleResult_t Gap_SendSmpKeys (deviceId_t deviceId, const gapSmpKeys_t ∗pKeys)
- bleResult_t Gap_RejectKeyExchangeRequest (deviceId_t deviceId)
- bleResult_t Gap_LeScRegeneratePublicKey (void)
- bleResult_t Gap_LeScValidateNumericValue (deviceId_t deviceId, bool_t valid)
- bleResult_t Gap_LeScGetLocalOobData (void)
- bleResult_t Gap_LeScSetPeerOobData (deviceId_t deviceId, const gapLeScOobData_t ∗pPeer↩ OobData)
- bleResult_t Gap_LeScSendKeypressNotification (deviceId_t deviceId, gapKeypressNotification_↩ t keypressNotification)
- bleResult_t Gap_ProvideLongTermKey (deviceId_t deviceId, const uint8_t ∗aLtk, uint8_t ltkSize)
- bleResult_t Gap_DenyLongTermKey (deviceId_t deviceId)
- bleResult_t Gap_LoadEncryptionInformation (deviceId_t deviceId, uint8_t ∗aOutLtk, uint8_t ∗p↩ OutLtkSize)
- bleResult_t Gap_SetLocalPasskey (uint32_t passkey)
- bleResult_t Gap_SetScanMode (gapScanMode_t scanMode, gapAutoConnectParams_t ∗pAuto↩ ConnectParams, gapConnectionCallback_t connCallback)
- bleResult_t Gap_StartScanning (const gapScanningParameters_t ∗pScanningParameters, gap↩ ScanningCallback_t scanningCallback, gapFilterDuplicates_t enableFilterDuplicates, uint16_t duration, uint16_t period)
- bleResult_t Gap_StopScanning (void)
- bleResult_t Gap_Connect (const gapConnectionRequestParameters_t ∗pParameters, gap↩

ConnectionCallback_t connCallback)
- bleResult_t Gap_Disconnect (deviceId_t deviceId)
- bleResult_t Gap_SaveCustomPeerInformation (deviceId_t deviceId, const uint8_t ∗aInfo, uint16_t offset, uint16_t infoSize)
- bleResult_t Gap_LoadCustomPeerInformation (deviceId_t deviceId, uint8_t ∗aOutInfo, uint16_↩t offset, uint16_t infoSize)
- bleResult_t Gap_CheckIfBonded (deviceId_t deviceId, bool_t ∗pOutIsBonded, uint8_t ∗pOut↩NvmIndex)
- bleResult_t Gap_ReadWhiteListSize (void)
- bleResult_t Gap_ClearWhiteList (void)
- bleResult_t Gap_AddDeviceToWhiteList (bleAddressType_t addressType, const bleDevice↩Address_t address)
- bleResult_t Gap_RemoveDeviceFromWhiteList (bleAddressType_t addressType, const bleDevice↩Address_t address)
- bleResult_t Gap_ReadPublicDeviceAddress (void)
- bleResult_t Gap_CreateRandomDeviceAddress (const uint8_t ∗aIrk, const uint8_t ∗aRandomPart)
- bleResult_t Gap_SaveDeviceName (deviceId_t deviceId, const uchar_t ∗aName, uint8_t cName↩Size)
- bleResult_t Gap_GetBondedDevicesCount (uint8_t ∗pOutBondedDevicesCount)
- bleResult_t Gap_GetBondedDeviceName (uint8_t nvmIndex, uchar_t ∗aOutName, uint8_t max↩NameSize)
- bleResult_t Gap_RemoveBond (uint8_t nvmIndex)
- bleResult_t Gap_RemoveAllBonds (void)
- bleResult_t Gap_ReadRadioPowerLevel (gapRadioPowerLevelReadType_t txReadType, deviceId↩_t deviceId)
- bleResult_t Gap_SetTxPowerLevel (uint8_t powerLevel, bleTransmitPowerChannelType_↩t channelType)
- bleResult_t Gap_VerifyPrivateResolvableAddress (uint8_t nvmIndex, const bleDeviceAddress_t a↩Address)
- bleResult_t Gap_SetRandomAddress (const bleDeviceAddress_t aAddress)
- bleResult_t Gap_SetDefaultPairingParameters (const gapPairingParameters_t ∗pPairingParameters)
- bleResult_t Gap_UpdateConnectionParameters (deviceId_t deviceId, uint16_t intervalMin, uint16↩_t intervalMax, uint16_t slaveLatency, uint16_t timeoutMultiplier, uint16_t minCeLength, uint16↩_t maxCeLength)
- bleResult_t Gap_EnableUpdateConnectionParameters (deviceId_t deviceId, bool_t enable)
- bleResult_t Gap_UpdateLeDataLength (deviceId_t deviceId, uint16_t txOctets, uint16_t txTime)
- bleResult_t Gap_EnableHostPrivacy (bool_t enable, const uint8_t ∗aIrk)
- bleResult_t Gap_EnableControllerPrivacy (bool_t enable, const uint8_t ∗aOwnIrk, uint8_t peerId↩Count, const gapIdentityInformation_t ∗aPeerIdentities)
- bleResult_t Gap_SetPrivacyMode (uint8_t nvmIndex, blePrivacyMode_t privacyMode)
- bleResult_t Gap_ControllerTest (gapControllerTestCmd_t testCmd, uint8_t radioChannel, uint8_t txDataLength, gapControllerTestTxType_t txPayloadType)
- bleResult_t Gap_LeReadPhy (deviceId_t deviceId)
- bleResult_t Gap_LeSetPhy (bool_t defaultMode, deviceId_t deviceId, uint8_t allPhys, uint8_t tx↩Phys, uint8_t rxPhys, uint16_t phyOptions)
- bleResult_t Gap_ControllerEnhancedNotification (uint16_t eventType, deviceId_t deviceId)
- bleResult_t Gap_LoadKeys (uint8_t nvmIndex, gapSmpKeys_t ∗pOutKeys, gapSmpKeyFlags_↩t ∗pOutKeyFlags, bool_t ∗pOutLeSc, bool_t ∗pOutAuth)
- bleResult_t Gap_SaveKeys (uint8_t nvmIndex, const gapSmpKeys_t ∗pKeys, bool_t leSc, bool_t auth)
- bleResult_t Gap_SetChannelMap (const bleChannelMap_t channelMap)
- bleResult_t Gap_ReadChannelMap (deviceId_t deviceId)

- bleResult_t Gap_SetExtAdvertisingParameters (gapExtAdvertisingParameters_t ∗pAdvertising↩ Parameters)
- bleResult_t Gap_SetExtAdvertisingData (uint8_t handle, gapAdvertisingData_t ∗pAdvertisingData, gapScanResponseData_t ∗pScanResponseData)
- bleResult_t Gap_StartExtAdvertising (gapAdvertisingCallback_t advertisingCallback, gap↩ ConnectionCallback_t connectionCallback, uint8_t handle, uint16_t duration, uint8_t maxExt↩ AdvEvents)
- bleResult_t Gap_StopExtAdvertising (uint8_t handle)
- bleResult_t Gap_RemoveAdvSet (uint8_t handle)
- bleResult_t Gap_SetPeriodicAdvParameters (gapPeriodicAdvParameters_t ∗pAdvertising↩ Parameters)
- bleResult_t Gap_SetPeriodicAdvertisingData (uint8_t handle, gapAdvertisingData_t ∗p↩ AdvertisingData)
- bleResult_t Gap_StartPeriodicAdvertising (uint8_t handle)
- bleResult_t Gap_StopPeriodicAdvertising (uint8_t handle)
- bleResult_t Gap_UpdatePeriodicAdvList (gapPeriodicAdvListOperation_t operation, bleAddress↩ Type_t addrType, uint8_t ∗pAddr, uint8_t SID)
- bleResult_t Gap_PeriodicAdvCreateSync (gapPeriodicAdvSyncReq_t ∗pReq)
- bleResult_t Gap_PeriodicAdvTerminateSync (uint16_t syncHandle)
- bleResult_t Gap_ResumeLeScStateMachine (computeDhKeyParam_t ∗pData)

3.2 Data Structure Documentation

3.2.1 struct gapSmpKeys_t

Structure containing the SMP information exchanged during pairing.

Data Fields

uint8_t	cLtkSize	Encryption Key Size. If aLtk is NULL, this is ignored.
uint8_t ∗	aLtk	Long Term (Encryption) Key. NULL if LTK is not distributed, else size is given by cLtkSize.
uint8_t ∗	aIrk	Identity Resolving Key. NULL if aIrk is not distributed.
uint8_t ∗	aCsrk	Connection Signature Resolving Key. NULL if aCsrk is not distributed.
uint8_t	cRandSize	Size of RAND; usually equal to gcMaxRandSize_d. If aLtk is N↩ ULL, this is ignored.
uint8_t ∗	aRand	RAND value used to identify the LTK. If aLtk is NULL, this is ignored.
uint16_t	ediv	EDIV value used to identify the LTK. If aLtk is NULL, this is ignored.
bleAddress↩ Type_t	addressType	Public or Random address. If aAddress is NULL, this is ignored.

uint8_t *	aAddress	Device Address. NULL if address is not distributed. If aIrk is N↩ULL, this is ignored.

3.2.2 struct gapSecurityRequirements_t

Security Requirements structure for a Device, a Service or a Characteristic.

Data Fields

gapSecurity↩ModeAnd↩Level_t	securityMode↩Level	Security mode and level.
bool_t	authorization	Authorization required.
uint16_t	minimum↩Encryption↩KeySize	Minimum encryption key (LTK) size. Ignored if gSecurityMode↩_1_Level_4_c is required (set to gMaxEncryptionKeySize_d automatically)

3.2.3 struct gapServiceSecurityRequirements_t

Service Security Requirements.

Data Fields

uint16_t	serviceHandle	Handle of the Service declaration in the GATT Database.
gapSecurity↩Requirements↩_t	requirements	Requirements for all attributes in this service.

3.2.4 struct gapDeviceSecurityRequirements_t

Device Security - Master Security Requirements + Service Security Requirements.

Data Fields

gapSecurity↩Requirements↩_t *	pMaster↩Security↩Requirements	Security requirements added to all services.

uint8_t	cNumServices	Number of service-specific requirements; must be less than or equal to gcGapMaxServiceSpecificSecurityRequirements_c.
gapService↩Security↩Requirements↩_t *	aService↩Security↩Requirements	Array of service-specific requirements.

3.2.5 struct gapHandleList_t

List of Attribute Handles for authorization lists.

Data Fields

uint8_t	cNumHandles	Number of handles in this list.
uint16_t	aHandles[gc↩GapMax↩Authorization↩Handles_c]	List of handles.

3.2.6 struct gapConnectionSecurityInformation_t

Connection Security Information structure.

Data Fields

bool_t	authenticated	TRUE if pairing was performed with MITM protection.
gapHandle↩List_t	authorizedTo↩Read	List of handles the peer has been authorized to read.
gapHandle↩List_t	authorizedTo↩Write	List of handles the peer has been authorized to write.

3.2.7 struct gapPairingParameters_t

Pairing parameters structure for the Gap_Pair and Gap_AcceptPairingRequest APIs.

Data Fields

bool_t	withBonding	TRUE if this device is able to and wants to bond after pairing, F↩ALSE otherwise.

gapSecurity↩ModeAnd↩Level_t	securityMode↩AndLevel	The desired security mode-level.
uint8_t	max↩Encryption↩KeySize	Maximum LTK size supported by the device.
gapIo↩Capabilities_t	localIo↩Capabilities	I/O capabilities used to determine the pairing method.
bool_t	oobAvailable	TRUE if this device has Out-of-Band data that can be used for authenticated pairing. FALSE otherwise.
gapSmpKey↩Flags_t	centralKeys	Indicates the SMP keys to be distributed by the Central.
gapSmpKey↩Flags_t	peripheralKeys	Indicates the SMP keys to be distributed by the Peripheral.
bool_t	leSecure↩Connection↩Supported	Indicates if device supports LE Secure Connections pairing. Conflict if this is FALSE and securityModeAndLevel is gSecurity↩Mode_1_Level_4_c.
bool_t	useKeypress↩Notifications	Indicates if device supports Keypress Notification PDUs during Passkey Entry pairing. Conflict if this is TRUE and localIo↩Capabilities is set to gIoNone_c.

3.2.8 struct gapSlaveSecurityRequestParameters_t

Parameters of a Slave Security Request.

Data Fields

bool_t	bondAfter↩Pairing	TRUE if the Slave supports bonding.
bool_t	authentication↩Required	TRUE if the Slave requires authentication for MITM protection.

3.2.9 struct gapAdvertisingParameters_t

Advertising Parameters; for defaults see gGapDefaultAdvertisingParameters_d.

Data Fields

uint16_t	minInterval	Minimum desired advertising interval. Default: 1.28 s.
uint16_t	maxInterval	Maximum desired advertising interval. Default: 1.28 s.

bleAdvertisingType_t	advertisingType	Advertising type. Default: connectable undirected.
bleAddressType_t	ownAddressType	Indicates whether the advertising address is the public address (BD_ADDR) or the random address (set by Gap_SetRandomAddress). Default: public address. If Controller Privacy is enabled, this parameter is irrelevant as Private Resolvable Addresses are always used.
bleAddressType_t	peerAddressType	Address type of the peer; only used in directed advertising and Enhanced Privacy.
bleDeviceAddress_t	peerAddress	Address of the peer; same as above.
gapAdvertisingChannelMapFlags_t	channelMap	Bit mask indicating which of the three advertising channels are used. Default: all three.
gapAdvertisingFilterPolicy_t	filterPolicy	Indicates whether the connect and scan requests are filtered using the White List. Default: does not use White List (process all).

3.2.10 struct gapExtAdvertisingParameters_t

Extended Advertising Parameters; for defaults see gGapDefaultExtAdvertisingParameters_d.

Data Fields

uint8_t	SID	ID of the advertising set chosen by application. Shall be lower than gBleExtAdvMaxSetId_c
uint8_t	handle	ID of the advertising set handled by controller. Shall be lower than gMaxAdvSets_c
uint32_t	minInterval	Minimum desired advertising interval. Shall be at least equal or higher than gGapExtAdvertisingIntervalRangeMinimum_c
uint32_t	maxInterval	Maximum desired advertising interval. Shall be higher than gGapExtAdvertisingIntervalRangeMinimum_c and higher than minInterval
bleAddressType_t	ownAddressType	Indicates whether the advertising address is the public address (BD_ADDR) or the random address (set by Gap_SetRandomAddress). Default: public address. If Controller Privacy is enabled, this parameter is irrelevant as Private Resolvable Addresses are always used.

bleDevice↩Address_t	ownRandom↩Addr	The random address used for advertising on the current handle.
bleAddress↩Type_t	peerAddress↩Type	Address type of the peer; only used in directed advertising and Enhanced Privacy.
bleDevice↩Address_t	peerAddress	Address of the peer; same as above.
gap↩Advertising↩ChannelMap↩Flags_t	channelMap	Bit mask indicating which of the three advertising channels are used for primary advertising.
gap↩Advertising↩FilterPolicy_t	filterPolicy	Indicates whether the connect and scan requests are filtered using the White List.
bleAdv↩Request↩Properties_t	extAdv↩Properties	Type of advertising event.
int8_t	txPower	The maximum power level at which the adv packets are to be transmitted. The Controller shall choose a power level lower than or equal to the one specified by the Host. Valid range: -127 to 20
gapLePhy↩Mode_t	primaryPHY	The PHY on which the advertising packets are transmitted (1M or Coded PHY). Used for sending ADV_EXT_IND
gapLePhy↩Mode_t	secondaryPHY	The PHY used for sending AUX_ADV_IND PDU. Used only for Extended Advertising Events
uint8_t	secondary↩AdvMaxSkip	Maximum number of advertising events that can be skipped before the AUX_ADV_IND can be sent.
bool_t	enable↩ScanReq↩Notification	Indicates whether the Controller shall send notifications upon the receipt of a scan request PDU.

3.2.11 struct gapPeriodicAdvParameters_t

Periodic Advertising Parameters; for defaults see gGapDefaultPeriodicAdvParameters_d.

Data Fields

uint8_t	handle	ID of the advertising set handled by controller. Shall be lower than gMaxAdvSets_c
bool_t	addTxPower↩InAdv	Set this option to include the Tx power in advertising packet.

uint16_t	minInterval	Minimum advertising interval for periodic advertising.
uint16_t	maxInterval	Maximum advertising interval for periodic advertising. Should be different and higher than minInterval.

3.2.12 struct gapScanningParameters_t

Scanning parameters; for defaults see gGapDefaultScanningParameters_d.

Data Fields

bleScanType←_t	type	Scanning type. Default: passive.
uint16_t	interval	Scanning interval. Default: 10 ms.
uint16_t	window	Scanning window. Default: 10 ms.
bleAddress←Type_t	ownAddress←Type	Indicates whether the address used in scan requests is the public address (BD_ADDR) or the random address (set by Gap_Set←RandomAddress). Default: public address. If Controller Privacy is enabled, this parameter is irrelevant as Private Resolvable Addresses are always used.
bleScanning←FilterPolicy_t	filterPolicy	Indicates whether the advertising packets are filtered using the White List. Default: does not use White List (scan all).
gapLePhy←Flags_t	scanningPHYs	Indicates the PHYs on which the advertising packets should be received on the primary advertising channel.

3.2.13 struct gapPeriodicAdvSyncReq_t

Periodic Advertising Sync Request parameters.

Data Fields

gapCreate←SyncReq←FilterPolicy_t	filterPolicy	Indicates whether the periodic advertiser list is used or listen for a single device described by the following parameters. When the periodic advertiser list is used, the following parameters are ignored.
uint8_t	SID	The SID advertised by the periodic advertiser in the ADI field.
bleAddress←Type_t	peerAddress←Type	Periodic advertiser's address type (Public or Random)
bleDevice←Address_t	peerAddress	Periodic advertiser's address.

uint16_t	skipCount	The number of consecutive periodic advertising packets that the receiver may skip after successfully receiving a periodic advertising packet.
uint16_t	timeout	The maximum permitted time between successful receives. If this time is exceeded, synchronization is lost.

3.2.14 struct gapConnectionRequestParameters_t

Connection request parameter structure to be used in the Gap_Connect function; for API-defined defaults, use gGapDefaultConnectionRequestParameters_d.

Data Fields

uint16_t	scanInterval	Scanning interval. Default: 10 ms.
uint16_t	scanWindow	Scanning window. Default: 10 ms.
bleInitiator↩FilterPolicy_t	filterPolicy	Indicates whether the connection request is issued for a specific device or for all the devices in the White List. Default: specific device.
bleAddress↩Type_t	ownAddress↩Type	Indicates whether the address used in connection requests is the public address (BD_ADDR) or the random address (set by Gap_↩SetRandomAddress). Default: public address.
bleAddress↩Type_t	peerAddress↩Type	When connecting to a specific device (see filterPolicy), this indicates that device's address type. Default: public address.
bleDevice↩Address_t	peerAddress	When connecting to a specific device (see filterPolicy), this indicates that device's address.
uint16_t	connInterval↩Min	The minimum desired connection interval. Default: 100 ms.
uint16_t	connInterval↩Max	The maximum desired connection interval. Default: 200 ms.
uint16_t	connLatency	The desired connection latency (the maximum number of consecutive connection events the Slave is allowed to ignore). Default: 0.
uint16_t	supervision↩Timeout	The maximum time interval between consecutive over-the-air packets; if this timer expires, the connection is dropped. Default: 10 s.
uint16_t	connEvent↩LengthMin	The minimum desired connection event length. Default: 0 ms.
uint16_t	connEvent↩LengthMax	The maximum desired connection event length. Default: maximum possible, ~41 s. (lets the Controller decide).

bool_t	usePeer↩Identity↩Address	If Controller Privacy is enabled and this parameter is TRUE, the address defined in the peerAddressType and peerAddress is an identity address. Otherwise, it is a device address.
gapLePhy↩Flags_t	initiatingPHYs	Indicates the PHY on which the advertising packets should be received on the primary advertising channel and the PHY for which connection parameters have been specified.

3.2.15 struct gapConnectionParameters_t

Connection parameters as received in the gConnEvtConnected_c connection event.

Data Fields

uint16_t	connInterval	Interval between connection events.
uint16_t	connLatency	Number of consecutive connection events the Slave may ignore.
uint16_t	supervision↩Timeout	The maximum time interval between consecutive over-the-air packets; if this timer expires, the connection is dropped.
bleMaster↩Clock↩Accuracy_t	masterClock↩Accuracy	Accuracy of master's clock, allowing for frame detection optimizations.

3.2.16 struct gapAdStructure_t

Definition of an AD Structure as contained in Advertising and Scan Response packets.

An Advertising or Scan Response packet contains several AD Structures.

Data Fields

uint8_t	length	Total length of the [adType + aData] fields. Equal to 1 + length↩Of(aData).
gapAdType_t	adType	AD Type of this AD Structure.
uint8_t *	aData	Data contained in this AD Structure; length of this array is equal to (gapAdStructure_t.length - 1).

3.2.17 struct gapAdvertisingData_t

Advertising Data structure : a list of several gapAdStructure_t structures.

Data Fields

uint8_t	cNumAd↩ Structures	Number of AD Structures.
gapAd↩ Structure_t *	aAdStructures	Array of AD Structures.

3.2.18 struct gapExtScanNotification_t

Data Fields

uint8_t	handle	Advertising Handle.
bleAddress↩ Type_t	scannerAddr↩ Type	Scanner device's address type.
bleDevice↩ Address_t	aScannerAddr	Scanner device's address.
bool_t	scannerAddr↩ Resolved	Whether the address corresponds to Resolved Private Address.

3.2.19 struct gapAdvertisingSetTerminated_t

Data Fields

bleResult_t	status	Status of advertising set termination.
uint8_t	handle	Advertising Handle.
deviceId_t	deviceId	Valid only if the advertising ended with a connection.
uint8_t	num↩ Completed↩ ExtAdvEvents	Number of advertising events sent by Controller.

3.2.20 struct gapAdvertisingEvent_t

Advertising event structure: type + data.

Data Fields

gap↩ Advertising↩ EventType_t	eventType	Event type.

union gap↩ Advertising↩ Event_t	eventData	Event data, to be interpreted according to gapAdvertisingEvent_↩ t.eventType.

3.2.21 union gapAdvertisingEvent_t.eventData

Data Fields

bleResult_t	failReason	Event data for gAdvertisingCommandFailed_c event type: reason of failure to enable or disable advertising.
gapExtScan↩ Notification_t	scan↩ Notification	Event data for gExtScanNotification_c event type: Scan Request Received Event.
gap↩ Advertising↩ Set↩ Terminated_t	advSet↩ Terminated	Event received when advertising in a given advertising set has stopped.

3.2.22 struct gapScannedDevice_t

Scanned device information structure, obtained from LE Advertising Reports.

Data Fields

bleAddress↩ Type_t	addressType	Device's advertising address type.
bleDevice↩ Address_t	aAddress	Device's advertising address.
int8_t	rssi	RSSI on the advertising channel; may be compared to the TX power contained in the AD Structure of type gAdTxPowerLevel_c to estimate distance from the advertiser.
uint8_t	dataLength	Length of the advertising or scan response data.
uint8_t *	data	Advertising or scan response data.
ble↩ Advertising↩ ReportEvent↩ Type_t	advEventType	Advertising report type, indicating what type of event generated this data (advertising, scan response).

bool_t	directRpaUsed	TRUE if directed advertising with Resolvable Private Address as Direct Address was detected while Enhanced Privacy is enabled.
bleDevice↩ Address_t	directRpa	Resolvable Private Address set as Direct Address for directed advertising. Valid only when directRpaUsed is TRUE.
bool_t	advertising↩ Address↩ Resolved	If this is TRUE, the address contained in the addressType and a↩ Address fields is the identity address of a resolved RPA from the Advertising Address field. Otherwise, the address from the respective fields is the public or random device address contained in the Advertising Address field.

3.2.23 struct gapExtScannedDevice_t

Data Fields

bleAddress↩ Type_t	addressType	Device's advertising address type.
bleDevice↩ Address_t	aAddress	Device's advertising address.
uint8_t	SID	Advertising set id.
bool_t	advertising↩ Address↩ Resolved	If this is TRUE, the address contained in the addressType and a↩ Address fields is the identity address of a resolved RPA from the Advertising Address field. Otherwise, the address from the respective fields is the public or random device address contained in the Advertising Address field.
bleAdv↩ ReportEvent↩ Properties_t	advEvent↩ Properties	Advertising report properties, indicating what type of event generated this data (advertising, scan response).
int8_t	rssi	RSSI on the advertising channel; may be compared to the TX power contained in the AD Structure of type gAdTxPowerLevel_c to estimate distance from the advertiser.
int8_t	txPower	The Tx power level of the advertiser.
uint8_t	primaryPHY	Advertiser PHY for primary channel.
uint8_t	secondaryPHY	Advertiser PHY for secondary channel.
uint16_t	periodicAdv↩ Interval	Interval of the periodic advertising. Zero if not periodic advertising.
bool_t	directRpaUsed	TRUE if directed advertising with Resolvable Private Address as Direct Address was detected while Enhanced Privacy is enabled.

bleAddress↩Type_t	directRpaType	Address type for directed advertising. Valid only when directRpa↩Used is TRUE.
bleDevice↩Address_t	directRpa	Resolvable Private Address set as Direct Address for directed advertising. Valid only when directRpaUsed is TRUE.
uint16_t	dataLength	Length of the advertising or scan response data.
uint8_t *	pData	Advertising or scan response data.

3.2.24 struct gapPeriodicScannedDevice_t

Data Fields

uint16_t	syncHandle	Sync Handle.
int8_t	txPower	The Tx power level of the advertiser.
int8_t	rssi	RSSI on the advertising channel; may be compared to the TX power contained in the AD Structure of type gAdTxPowerLevel_c to estimate distance from the advertiser.
uint16_t	dataLength	Length of the advertising or scan response data.
uint8_t *	pData	Advertising or scan response data.

3.2.25 struct gapSyncEstbEventData_t

Data Fields

bleResult_t	status	Status of the Sync Established Event.
uint16_t	syncHandle	Sync Handle.

3.2.26 struct gapSyncLostEventData_t

Data Fields

uint16_t	syncHandle	Sync Handle.

3.2.27 struct gapScanningEvent_t

Scanning event structure: type + data.

Data Fields

gapScanning↩ EventType_t	eventType	Event type.
union gapScanning↩ Event_t	eventData	Event data, to be interpreted according to gapScanningEvent_t.↩ eventType.

3.2.28 union gapScanningEvent_t.eventData

Data Fields

bleResult_t	failReason	Event data for gScanCommandFailed_c or gPeriodicAdvSync↩ Established_c event type: reason of failure to enable/disable scanning or to establish sync.
gapScanned↩ Device_t	scannedDevice	Event data for gDeviceScanned_c event type: scanned device information.
gapExt↩ Scanned↩ Device_t	extScanned↩ Device	Event data for gExtDeviceScanned_c event type: extended scanned device information.
gapPeriodic↩ Scanned↩ Device_t	periodic↩ ScannedDevice	
gapSyncEstb↩ EventData_t	syncEstb	Event data for gPeriodicAdvSyncEstablished_c event type: Sync handle information for the application.
gapSyncLost↩ EventData_t	syncLost	Event data for gPeriodicAdvSyncLost_c event type: Sync handle information for the application.

3.2.29 struct gapConnectedEvent_t

Event data structure for the gConnEvtConnected_c event.

Data Fields

gap↩ Connection↩ Parameters_t	conn↩ Parameters	Connection parameters established by the Controller.
bleAddress↩ Type_t	peerAddress↩ Type	Connected device's address type.
bleDevice↩ Address_t	peerAddress	Connected device's address.

bool_t	peerRpa↩ Resolved	If this is TRUE, the address defined by peerAddressType and peerAddress is the identity address of the peer, and the peer used an RPA that was resolved by the Controller and is contained in the peerRpa field. Otherwise, it is a device address. This parameter is irrelevant if Controller Privacy is not enabled.
bleDevice↩ Address_t	peerRpa	Peer Resolvable Private Address if Controller Privacy is active and peerRpaResolved is TRUE.
bool_t	localRpaUsed	If this is TRUE, the Controller has used an RPA contained in the localRpa field. This parameter is irrelevant if Controller Privacy is not enabled.
bleDevice↩ Address_t	localRpa	Local Resolvable Private Address if Controller Privacy is active and localRpaUsed is TRUE.
bleLl↩ Connection↩ Role_t	connectionRole	Connection Role - master or slave.

3.2.30 struct gapKeyExchangeRequestEvent_t

Event data structure for the gConnEvtKeyExchangeRequest_c event.

Data Fields

gapSmpKey↩ Flags_t	requestedKeys	Mask identifying the keys being requested.
uint8_t	requestedLtk↩ Size	Requested size of the encryption key.

3.2.31 struct gapKeysReceivedEvent_t

Event data structure for the gConnEvtKeysReceived_c event.

Data Fields

gapSmpKeys↩ _t *	pKeys	The SMP keys distributed by the peer.

3.2.32 struct gapAuthenticationRejectedEvent_t

Event data structure for the gConnEvtAuthenticationRejected_c event.

Data Fields

gapAuthenticationRejectReason_t	rejectReason	Slave's reason for rejecting the authentication.

3.2.33 struct gapPairingCompleteEvent_t

Event data structure for the gConnEvtPairingComplete_c event.

Data Fields

bool_t	pairingSuccessful	TRUE if pairing succeeded, FALSE otherwise.
union gapPairingCompleteEvent_t	pairingCompleteData	Information of completion, selected upon the value of gapPairingCompleteEvent_t.pairingSuccessful.

3.2.34 union gapPairingCompleteEvent_t.pairingCompleteData

Data Fields

bool_t	withBonding	If pairingSuccessful is TRUE, this indicates whether the devices bonded.
bleResult_t	failReason	If pairingSuccessful is FALSE, this contains the reason of failure.

3.2.35 struct gapLongTermKeyRequestEvent_t

Event data structure for the gConnEvtLongTermKeyRequest_c event.

Data Fields

uint16_t	ediv	The Encryption Diversifier, as defined by the SMP.
uint8_t	aRand[gcSmpMaxRandSize_c]	The Random number, as defined by the SMP.

uint8_t	randSize	Usually equal to gcMaxRandSize_d.

3.2.36 struct gapEncryptionChangedEvent_t

Event data structure for the gConnEvtEncryptionChanged_c event.

Data Fields

bool_t	new↩ Encryption↩ State	TRUE if link has been encrypted, FALSE if encryption was paused or removed.

3.2.37 struct gapDisconnectedEvent_t

Event data structure for the gConnEvtDisconnected_c event.

Data Fields

gap↩ Disconnection↩ Reason_t	reason	Reason for disconnection.

3.2.38 struct gapConnParamsUpdateReq_t

Event data structure for the gConnEvtParameterUpdateRequest_c event.

Data Fields

uint16_t	intervalMin	Minimum interval between connection events.
uint16_t	intervalMax	Maximum interval between connection events.
uint16_t	slaveLatency	Number of consecutive connection events the Slave may ignore.
uint16_t	timeout↩ Multiplier	The maximum time interval between consecutive over-the-air packets; if this timer expires, the connection is dropped.

3.2.39 struct gapConnParamsUpdateComplete_t

Event data structure for the gConnEvtParameterUpdateComplete_c event.

Data Fields

bleResult_t	status	
uint16_t	connInterval	Interval between connection events.
uint16_t	connLatency	Number of consecutive connection events the Slave may ignore.
uint16_t	supervision↩ Timeout	The maximum time interval between consecutive over-the-air packets; if this timer expires, the connection is dropped.

3.2.40 struct gapConnLeDataLengthChanged_t

Event data structure for the gConnEvtLeDataLengthChanged_c event.

Data Fields

uint16_t	maxTxOctets	The maximum number of payload octets in a Link Layer Data Channel PDU to transmit on this connection.
uint16_t	maxTxTime	The maximum time that the local Controller will take to send a Link Layer Data Channel PDU on this connection.
uint16_t	maxRxOctets	The maximum number of payload octets in a Link Layer Data Channel PDU to receive on this connection.
uint16_t	maxRxTime	The maximum time that the local Controller will take to receive a Link Layer Data Channel PDU on this connection.

3.2.41 struct gapConnectionEvent_t

Connection event structure: type + data.

Data Fields

gap↩ Connection↩ EventType_t	eventType	Event type.
union gap↩ Connection↩ Event_t	eventData	Event data, to be interpreted according to gapConnectionEvent_↩ t.eventType.

3.2.42 union gapConnectionEvent_t.eventData

Data Fields

gap↩ Connected↩ Event_t	connected↩ Event	Data for gConnEvtConnected_c: information about the connection parameters.

gapPairing↩Parameters_t	pairingEvent	Data for gConnEvtPairingRequest_c, gConnEvtPairingResponse↩_c: pairing parameters.
gap↩Authentication↩Rejected↩Event_t	authentication↩RejectedEvent	Data for gConnEvtAuthenticationRejected_c: reason for rejection.
gapSlave↩Security↩Request↩Parameters_t	slaveSecurity↩RequestEvent	Data for gConnEvtSlaveSecurityRequest_c: Slave's security requirements.
gapKey↩Exchange↩RequestEvent↩_t	keyExchange↩RequestEvent	Data for gConnEvtKeyExchangeRequest_c: mask indicating the keys that were requested by the peer.
gapKeys↩Received↩Event_t	keysReceived↩Event	Data for gConnEvtKeysReceived_c: the keys received from the peer.
gapPairing↩Complete↩Event_t	pairing↩CompleteEvent	Data for gConnEvtPairingComplete_c: fail reason or (if successful) bonding state.
gapLong↩TermKey↩RequestEvent↩_t	longTermKey↩RequestEvent	Data for gConnEvtLongTermKeyRequest_c: encryption diversifier and random number.
gap↩Encryption↩Changed↩Event_t	encryption↩ChangedEvent	Data for gConnEvtEncryptionChanged_c: new encryption state.
gap↩Disconnected↩Event_t	disconnected↩Event	Data for gConnEvtDisconnected_c: reason for disconnection.
int8_t	rssi_dBm	Data for gConnEvtRssiRead_c: value of the RSSI in dBm.
int8_t	txPowerLevel↩_dBm	Data for gConnEvtTxPowerLevelRead_c: value of the TX power.
bleResult_t	failReason	Data for gConnEvtPowerReadFailure_c: reason for power reading failure.
uint32_t	passkeyFor↩Display	
gapConn↩Params↩UpdateReq_t	connection↩UpdateRequest	Data for gConnEvtParameterUpdateRequest_c: connection parameters update.

gapConn↩Params↩Update↩Complete_t	connection↩Update↩Complete	Data for gConnEvtParameterUpdateComplete_c: connection parameters update.
gapConnLe↩DataLength↩Changed_t	leDataLength↩Changed	Data for gConnEvtLeDataLengthChanged_c: new data length parameters.
gapKeypress↩Notification_t	incoming↩Keypress↩Notification	
uint32_t	numericValue↩ForDisplay	
bleChannel↩Map_t	channelMap	Data for gConnEvtChannelMapRead_c: channel map read from the Controller.

3.2.43 struct gapIdentityInformation_t

Identity Information structure definition.

Data Fields

bleIdentity↩Address_t	identity↩Address	Identity Address - Public or Random Static.
uint8_t	irk[gcSmpIrk↩Size_c]	Identity Resolving Key - must not be all-zero if Network Privacy is used.
blePrivacy↩Mode_t	privacyMode	Privacy mode to be used for the entry on the resolving list.

3.2.44 struct gapAutoConnectParams_t

Parameters for the Auto Connect Scan Mode.

Data Fields

uint8_t	cNum↩Addresses	Number of device addresses to automatically connect to.
bool_t	writeInWhite↩List	If set to TRUE, the device addresses are written in the White List before scanning is enabled.
gap↩Connection↩Request↩Parameters_t *	aAuto↩ConnectData	The array of connection request parameters, of size equal to c↩NumAddresses.

3.3 Macro Definition Documentation

3.3.1 #define Gap_AddSecurityModesAndLevels(*modeLevelA, modeLevelB*)

Macro used to combine two security mode-levels.

Parameters

in	*mode↩ LevelA,mode↩ LevelB*	The two security mode-levels.

Returns

The resulting security mode-level.

Remarks

This macro is useful when two different security requirements must be satisfied at the same time, such as a device master security requirement and a service-specific security requirement.

3.3.2 #define Gap_CancelInitiatingConnection( )

Macro used to cancel a connection initiated by Gap_Connect(...).

Returns

gBleSuccess_c or error.

Remarks

This macro can only be used for a connection that has not yet been established, such as the "gConn↩ EvtConnected_c" has not been received. For example, call this when a connection request has timed out. Application should listen for gCreateConnectionCanceled_c generic event.

3.3.3 #define Gap_ReadAdvertisingTxPowerLevel( )

Macro used to read the radio transmitter power when advertising.

Returns

gBleSuccess_c or error.

Remarks

The result is contained in the gAdvTxPowerLevelRead_c generic event.

3.3.4 #define Gap_ReadRssi(*deviceId*)

Macro used to read the RSSI of a radio connection.

Parameters

in	*deviceId*	Device ID identifying the radio connection.

Returns

gBleSuccess_c or error.

Remarks

The result is contained in the gConnEvtRssiRead_c connection event. The RSSI value is a signed byte, and the unit is dBm. If the RSSI cannot be read, the gConnEvtPowerReadFailure_c connection event is generated.

3.3.5 #define Gap_ReadTxPowerLevelInConnection(*deviceId*)

Macro used to read the radio transmitting power level of a radio connection.

Parameters

in	*deviceId*	Device ID identifying the radio connection.

Returns

gBleSuccess_c or error.

Remarks

The result is contained in the gConnEvtTxPowerLevelRead_c connection event. If the TX Power cannot be read, the gConnEvtPowerReadFailure_c connection event is generated.

3.3.6 #define gCancelOngoingInitiatingConnection_d

Use this value as a parameter to the Gap_Disconnect(deviceId) function to cancel any ongoing connection initiation, for example if the connection has timed out.

3.3.7 #define gMode_2_Mask_d

Mask to check if a Security Mode-and-Level is Mode 2.

3.3.8 #define getSecurityLevel(*modeLevel*)

Extracts the security level (see gapSecurityLevel_t) from the combined security mode-level (gapSecurity↩
ModeAndLevel_t).

3.3.9 #define getSecurityMode(*modeLevel*)

Extracts the security mode (see gapSecurityMode_t) from the combined security mode-level (gap↩
SecurityModeAndLevel_t).

3.3.10 #define gDefaultEncryptionKeySize_d

The default (minimum) value for the LTK size.

3.3.11 #define gMaxEncryptionKeySize_d

The maximum value for the LTK size.

3.3.12 #define gGapDefaultDeviceSecurity_d

The default value for the Device Security (no requirements)

3.3.13 #define gGapDefaultSecurityRequirements_d

The default value for a Security Requirement.

3.3.14 #define gGapAdvertisingIntervalRangeMinimum_c

Minimum advertising interval (20 ms)

3.3.15 #define gGapAdvertisingIntervalDefault_c

Default advertising interval (1.28 s)

3.3.16 #define gGapAdvertisingIntervalRangeMaximum_c

Maximum advertising interval (10.24 s)

3.3.17 #define gGapExtAdvertisingIntervalRangeMinimum_c

Minimum extended advertising interval (20 ms)

3.3.18 #define gGapExtAdvertisingIntervalDefault_c

Default extended advertising interval (1.28 s)

3.3.19 #define gGapExtAdvertisingIntervalRangeMaximum_c

Maximum extended advertising interval (10485.76 s)

3.3.20 #define gGapPeriodicAdvIntervalRangeMinimum_c

Minimum periodic advertising interval (7.5 ms)

3.3.21 #define gGapPeriodicAdvIntervalDefault_c

Default periodic advertising interval (2.56 s)

3.3.22 #define gGapPeriodicAdvIntervalRangeMaximum_c

Maximum periodic advertising interval (81.91875 s)

3.3.23 #define gGapAdvertisingChannelMapDefault_c

Default Advertising Channel Map - all 3 channels are enabled.

3.3.24 #define gGapDefaultAdvertisingParameters_d

Default value for Advertising Parameters struct.

3.3.25 #define gGapDefaultExtAdvertisingParameters_d

Default value for Extended Advertising Parameters struct.

3.3.26 #define gGapDefaultPeriodicAdvParameters_d

Default value for Periodic Advertising Parameters struct.

3.3.27 #define gGapScanIntervalMin_d

Minimum scan interval (2.5 ms)

3.3.28 #define gGapScanIntervalDefault_d

Default scan interval (10 ms)

3.3.29 #define gGapScanIntervalMax_d

Maximum scan interval (10.24 s)

3.3.30 #define gGapScanWindowMin_d

Minimum scan window (2.5 ms)

3.3.31 #define gGapScanWindowDefault_d

Default scan window (10 ms)

3.3.32 #define gGapScanWindowMax_d

Maximum scan window (10.24 s)

3.3.33 #define gGapRssiMin_d

Minimum valid value for RSSI (dB)

3.3.34 #define gGapRssiMax_d

Maximum valid value for RSSI (dB)

3.3.35 #define gGapRssiNotAvailable_d

A special invalid value for the RSSI indicating that the measurement is not available.

3.3.36 #define gGapScanContinuously_d

Default value for Scanning duration - Scan continuously until explicitly disable.

3.3.37 #define gGapScanPeriodicDisabled_d

Default value for Scanning period - Periodic scanning disabled.

3.3.38 #define gGapDefaultScanningParameters_d

Default value for Scanning Parameters struct.

3.3.39 #define gGapConnIntervalMin_d

Minimum connection interval (7.5 ms)

3.3.40 #define gGapConnIntervalMax_d

Maximum connection interval (4 s)

3.3.41 #define gGapConnLatencyMin_d

Minimum connection latency value (0 - no connection event may be ignored)

3.3.42 #define gGapConnLatencyMax_d

Maximum connection latency value (499 connection events may be ignored)

3.3.43 #define gGapConnSuperTimeoutMin_d

Minimum supervision timeout (100 ms)

3.3.44 #define gGapConnSuperTimeoutMax_d

Maximum supervision timeout (32 s)

3.3.45 #define gGapConnEventLengthMin_d

Minimum value of the connection event length (0 ms)

3.3.46 #define gGapConnEventLengthMax_d

Maximum value of the connection event length (∼41 s)

3.3.47 #define gGapDefaultConnectionLatency_d

Default connection latency: 0.

3.3.48 #define gGapDefaultSupervisionTimeout_d

Default supervision timeout: 10s.

3.3.49 #define gGapDefaultMinConnectionInterval_d

Default minimum connection interval: 100ms.

3.3.50 #define gGapDefaultMaxConnectionInterval_d

Default maximum connection interval: 200ms.

3.3.51 #define gGapDefaultConnectionRequestParameters_d

The default value for the Connection Request Parameters structure.

3.3.52 #define gGapChSelAlgorithmNo2

"Channel Selection Algorithm #2 is used" value in LE Channel Selection Algorithm Event

3.3.53 #define gBlePeriodicAdvOngoingSyncCancelHandle

Sync handle value for which to call the create sync cancel command instead of terminate sync.

3.3.54 #define gGapInvalidSyncHandle

Sync handle used to differentiate extended advertising reports from periodic advertising reports.

3.3.55 #define gNone_c

Values of the AD Flags advertising data structure.
No information.

3.3.56 #define gLeLimitedDiscoverableMode_c

This device is in Limited Discoverable mode.

3.3.57 #define gLeGeneralDiscoverableMode_c

This device is in General Discoverable mode.

3.3.58 #define gBrEdrNotSupported_c

This device supports only Bluetooth Low Energy; no support for Classic Bluetooth.

3.3.59 #define gSimultaneousLeBrEdrCapableController_c

This device's Controller also supports Classic Bluetooth.

3.3.60 #define gSimultaneousLeBrEdrCapableHost_c

This device's Host also supports Classic Bluetooth.

3.3.61 #define gNoKeys_c

Flags indicating the Keys to be exchanged by the SMP during the key exchange phase of pairing.

No key can be distributed.

3.3.62 #define gLtk_c

Long Term Key.

3.3.63 #define gIrk_c

Identity Resolving Key.

3.3.64 #define gCsrk_c

Connection Signature Resolving Key.

3.3.65 #define gSecurityMode_1_c

LE Security Mode values for gapSecurityMode_t.

Mode 1 - Encryption required (except for Level 1).

3.3.66 #define gSecurityMode_2_c

Mode 2 - Data Signing required.

3.3.67 #define gSecurityLevel_NoSecurity_c

LE Security Level values for gapSecurityLevel_t.

No security (combined only with Mode 1).

3.3.68 #define gSecurityLevel_NoMitmProtection_c

Unauthenticated (no MITM protection).

3.3.69 #define gSecurityLevel_WithMitmProtection_c

Authenticated (MITM protection by PIN or OOB).

3.3.70 #define gSecurityLevel_LeSecureConnections_c

Authenticated with LE Secure Connections.

3.3.71 #define gSecurityMode_1_Level_1_c

Security Mode-and-Level definitions values for gapSecurityModeAndLevel_t.

Mode 1 Level 1 - No Security.

3.3.72 #define gSecurityMode_1_Level_2_c

Mode 1 Level 2 - Encryption without authentication.

3.3.73 #define gSecurityMode_1_Level_3_c

Mode 1 Level 3 - Encryption with authentication.

3.3.74 #define gSecurityMode_1_Level_4_c

Mode 1 Level 4 - Encryption with LE Secure Connections pairing.

3.3.75 #define gSecurityMode_2_Level_1_c

Mode 2 Level 1 - Data Signing without authentication.

3.3.76 #define gSecurityMode_2_Level_2_c

Mode 2 Level 2 - Data Signing with authentication.

3.3.77 #define gOobNotAvailable_c

Reason for rejecting the pairing request used by gapAuthenticationRejectReason_t.

These values are equal to the corresponding reasons from SMP. This device does not have the required OOB for authenticated pairing.

3.3.78 #define gIncompatibleIoCapabilities_c

The combination of I/O capabilities does not allow pairing with the desired level of security.

3.3.79 #define gPairingNotSupported_c

This device does not support pairing.

3.3.80 #define gLowEncryptionKeySize_c

The peer's encryption key size is too low for this device's required security level.

3.3.81 #define gUnspecifiedReason_c

The host has rejected the pairing for an unknown reason.

3.3.82 #define gRepeatedAttempts_c

This device is the target of repeated unsuccessful pairing attempts and does not allow further pairing attempts at the moment.

3.3.83 #define gLinkEncryptionFailed_c

Link could not be encrypted.

This reason may not be used by Gap_RejectPairing!

3.3.84 #define gIoDisplayOnly_c

I/O Capabilities as defined by the SMP used by gapIoCapabilities_t.

May display a PIN, no input.

3.3.85 #define gIoDisplayYesNo_c

May display a PIN and has a binary input (e.g., YES and NO buttons).

3.3.86 #define gIoKeyboardOnly_c

Has keyboard input, no display.

3.3.87 #define gIoNone_c

No input and no display.

3.3.88 #define gIoKeyboardDisplay_c

Has keyboard input and display.

3.4 Typedef Documentation

3.4.1 typedef uint8_t gapIoCapabilities_t

I/O Capabilities as defined by the SMP.

3.4.2 typedef uint8_t gapSmpKeyFlags_t

Flags indicating the Keys to be exchanged by the SMP during the key exchange phase of pairing.

3.4.3 typedef uint8_t gapSecurityMode_t

LE Security Mode.

3.4.4 typedef uint8_t gapSecurityLevel_t

LE Security Level.

3.4.5 typedef uint8_t gapSecurityModeAndLevel_t

Security Mode-and-Level definitions.

3.4.6 typedef uint8_t gapKeypressNotification_t

Keypress Notification Types.

3.4.7 typedef uint8_t gapAuthenticationRejectReason_t

Reason for rejecting the pairing request.

3.4.8 typedef uint8_t gapCreateSyncReqFilterPolicy_t

Create Sync Request Filter Policy values.

3.4.9 typedef uint8_t gapAdTypeFlags_t

Values of the AD Flags advertising data structure.

3.4.10 typedef gapAdvertisingData_t gapScanResponseData_t

Scan Response Data structure : a list of several gapAdStructure_t structures.

3.4.11 typedef uint8_t gapControllerTestTxType_t

Enumeration for Controller Transmitter Test payload types.

3.4.12 typedef bleResult_t gapDisconnectionReason_t

Disconnection reason alias - reasons are contained in HCI error codes.

3.4.13 typedef void(∗ gapAdvertisingCallback_t) (gapAdvertisingEvent_t ∗pAdvertisingEvent)

Advertising Callback prototype.

3.4.14 typedef void(∗ gapScanningCallback_t) (gapScanningEvent_t ∗pScanningEvent)

Scanning Callback prototype.

3.4.15 typedef void(∗ gapConnectionCallback_t) (deviceId_t deviceId, gapConnectionEvent_t ∗pConnectionEvent)

Connection Callback prototype.

3.5 Enumeration Type Documentation

3.5.1 enum gapRole_t

GAP Role of a BLE device.

Enumerator

gGapCentral_c	Central scans and connects to Peripherals.
gGapPeripheral_c	Peripheral advertises and connects to Centrals.
gGapObserver_c	Observer only scans and makes no connections.
gGapBroadcaster_c	Broadcaster only advertises and makes no connections.

3.5.2 enum gapKeypressNotification_tag

Enumerator

gKnPasskeyEntryStarted_c	Start of the Passkey Entry.
gKnPasskeyDigitStarted_c	Digit entered.
gKnPasskeyDigitErased_c	Digit erased.
gKnPasskeyCleared_c	Passkey cleared.
gKnPasskeyEntryCompleted_c	Passkey Entry completed.

3.5.3 enum gapScanMode_t

Scan Mode options; used as parameter for Gap_SetScanMode.

Enumerator

gDefaultScan_c	Reports all scanned devices to the application.
gLimitedDiscovery_c	Reports only devices in Limited Discoverable Mode, i.e., containing the Flags AD with the LE Limited Discoverable Flag set.

gGeneralDiscovery_c Reports only devices in General Discoverable Mode, i.e., containing the Flags AD with the LE General Discoverable Flag set.
gAutoConnect_c Automatically connects with devices with known addresses and does not report any scanned device to the application.

3.5.4 enum gapAdvertisingChannelMapFlags_t

Advertising Channel Map flags - setting a bit activates advertising on the respective channel.

Enumerator

gAdvChanMapFlag37_c Bit for channel 37.
gAdvChanMapFlag38_c Bit for channel 38.
gAdvChanMapFlag39_c Bit for channel 39.

3.5.5 enum gapAdvertisingFilterPolicy_t

Advertising Filter Policy values.

Enumerator

gProcessAll_c Default value: accept all connect and scan requests.
gProcessConnAllScanWL_c Accept all connect requests, but scan requests only from devices in White List.
gProcessScanAllConnWL_c Accept all scan requests, but connect requests only from devices in White List.
gProcessWhiteListOnly_c Accept connect and scan requests only from devices in White List.

3.5.6 enum gapFilterDuplicates_t

Enumerator

gGapDuplicateFilteringDisable_c Duplicate filtering disabled.
gGapDuplicateFilteringEnable_c Duplicate filtering enabled.
gGapDuplicateFilteringPeriodicEnable_c Duplicate filtering enabled, reset for each scan period.

3.5.7 enum gapCreateSyncReqFilterPolicy_tag

Enumerator

gUseCommandParameters_c Use the SID, peerAddressType, and peerAddress parameters to determine which advertiser to listen to.
gUsePeriodicAdvList_c Use the Periodic Advertiser List to determine which advertiser to listen to.

3.5.8 enum gapAdType_t

AD Type values as defined by Bluetooth SIG used when defining gapAdStructure_t structures for advertising or scan response data.

Enumerator

gAdFlags_c Defined by the Bluetooth SIG.
gAdIncomplete16bitServiceList_c Defined by the Bluetooth SIG.
gAdComplete16bitServiceList_c Defined by the Bluetooth SIG.
gAdIncomplete32bitServiceList_c Defined by the Bluetooth SIG.
gAdComplete32bitServiceList_c Defined by the Bluetooth SIG.
gAdIncomplete128bitServiceList_c Defined by the Bluetooth SIG.
gAdComplete128bitServiceList_c Defined by the Bluetooth SIG.
gAdShortenedLocalName_c Defined by the Bluetooth SIG.
gAdCompleteLocalName_c Defined by the Bluetooth SIG.
gAdTxPowerLevel_c Defined by the Bluetooth SIG.
gAdClassOfDevice_c Defined by the Bluetooth SIG.
gAdSimplePairingHashC192_c Defined by the Bluetooth SIG.
gAdSimplePairingRandomizerR192_c Defined by the Bluetooth SIG.
gAdSecurityManagerTkValue_c Defined by the Bluetooth SIG.
gAdSecurityManagerOobFlags_c Defined by the Bluetooth SIG.
gAdSlaveConnectionIntervalRange_c Defined by the Bluetooth SIG.
gAdServiceSolicitationList16bit_c Defined by the Bluetooth SIG.
gAdServiceSolicitationList32bit_c Defined by the Bluetooth SIG.
gAdServiceSolicitationList128bit_c Defined by the Bluetooth SIG.
gAdServiceData16bit_c Defined by the Bluetooth SIG.
gAdServiceData32bit_c Defined by the Bluetooth SIG.
gAdServiceData128bit_c Defined by the Bluetooth SIG.
gAdPublicTargetAddress_c Defined by the Bluetooth SIG.
gAdRandomTargetAddress_c Defined by the Bluetooth SIG.
gAdAppearance_c Defined by the Bluetooth SIG.
gAdAdvertisingInterval_c Defined by the Bluetooth SIG.
gAdLeDeviceAddress_c Defined by the Bluetooth SIG.
gAdLeRole_c Defined by the Bluetooth SIG.
gAdSimplePairingHashC256_c Defined by the Bluetooth SIG.

gAdSimplePairingRandomizerR256_c Defined by the Bluetooth SIG.
gAd3dInformationData_c Defined by the Bluetooth SIG.
gAdUniformResourceIdentifier_c Defined by the Bluetooth SIG.
gAdLeSupportedFeatures_c Defined by the Bluetooth SIG.
gAdChannelMapUpdateIndication_c Defined by the Bluetooth SIG.
gAdManufacturerSpecificData_c Defined by the Bluetooth SIG.

3.5.9 enum gapRadioPowerLevelReadType_t

Enumeration used by the Gap_ReadRadioPowerLevel function.

Enumerator

gTxPowerCurrentLevelInConnection_c Reading the instantaneous TX power level in a connection.

gTxPowerMaximumLevelInConnection_c Reading the maximum TX power level achieved during a connection.
gTxPowerLevelForAdvertising_c Reading the TX power on the advertising channels.
gRssi_c Reading the Received Signal Strength Indication in a connection.

3.5.10 enum gapControllerTestCmd_t

Enumeration for Controller Test commands.

Enumerator

gControllerTestCmdStartRx_c Start Receiver Test.
gControllerTestCmdStartTx_c Start Transmitter Test.
gControllerTestCmdEnd_c End Test.

3.5.11 enum gapControllerTestTxType_tag

Enumerator

gControllerTestTxPrbs9_c PRBS9 sequence 11111111100000111101
gControllerTestTxF0_c Repeated 11110000
gControllerTestTxAA_c Repeated 10101010
gControllerTestTxPrbs15_c PRBS15 sequence.
gControllerTestTxFF_c Repeated 11111111
gControllerTestTx00_c Repeated 00000000
gControllerTestTx0F_c Repeated 00001111
gControllerTestTx55_c Repeated 01010101

3.5.12 enum gapAdvertisingEventType_t

Advertising event type enumeration, as contained in the gapAdvertisingEvent_t.

Enumerator

gAdvertisingStateChanged_c Event received when advertising has been successfully enabled or disabled.

gAdvertisingCommandFailed_c Event received when advertising could not be enabled or disabled. Reason contained in gapAdvertisingEvent_t.eventData.failReason.

gExtAdvertisingStateChanged_c Event received when extended advertising has been successfully enabled or disabled.

gAdvertisingSetTerminated_c Event received when advertising in a given advertising set has stopped.

gExtScanNotification_c Event indicates that a SCAN_REQ PDU or an AUX_SCAN_REQ PDU has been received by the extended advertiser.

gPeriodicAdvertisingStateChanged_c Event received when periodic advertising has been successfully enabled or disabled.

3.5.13 enum gapScanningEventType_t

Scanning event type enumeration, as contained in the gapScanningEvent_t.

Enumerator

gScanStateChanged_c Event received when scanning had been successfully enabled or disabled, or a Scan duration time-out has occurred.

gScanCommandFailed_c Event received when scanning could not be enabled or disabled. Reason contained in gapScanningEvent_t.eventData.failReason.

gDeviceScanned_c Event received when an advertising device has been scanned. Device data contained in gapScanningEvent_t.eventData.scannedDevice.

gExtDeviceScanned_c Event received when an advertising device has been scanned. Device data contained in gapScanningEvent_t.eventData.extScannedDevice.

gPeriodicDeviceScanned_c Event received when an Periodic advertising device has been scanned. Device data contained in gapScanningEvent_t.eventData.periodicScannedDevice.

gPeriodicAdvSyncEstablished_c Event received when a sync with a periodic advertiser was established.

gPeriodicAdvSyncLost_c Event received when a sync with a periodic advertiser have been lost.

gPeriodicAdvSyncTerminated_c Event received when a sync with a periodic advertiser have been terminated.

3.5.14 enum gapConnectionEventType_t

Connection event type enumeration, as contained in the gapConnectionEvent_t.

Enumerator

gConnEvtConnected_c A connection has been established. Data in gapConnectionEvent_t.event↩ Data.connectedEvent.

gConnEvtPairingRequest_c A pairing request has been received from the peer Master. Data in gapConnectionEvent_t.eventData.pairingEvent.

gConnEvtSlaveSecurityRequest_c A Slave Security Request has been received from the peer Slave. Data in gapConnectionEvent_t.eventData.slaveSecurityRequestEvent.

gConnEvtPairingResponse_c A pairing response has been received from the peer Slave. Data in gapConnectionEvent_t.eventData.pairingEvent.

gConnEvtAuthenticationRejected_c A link encryption or pairing request has been rejected by the peer device. Data in gapConnectionEvent_t.eventData.authenticationRejectedEvent.

gConnEvtPasskeyRequest_c Peer Slave has requested a passkey (maximum 6 digit PIN) for the pairing procedure. Master should respond with Gap_EnterPasskey. Slave will not receive this event! Slave's application must call Gap_SetLocalPasskey before any connection.

gConnEvtOobRequest_c Out-of-Band data must be provided for the pairing procedure. Master or Slave should respond with Gap_ProvideOob.

gConnEvtPasskeyDisplay_c The pairing procedure requires this Slave to display the passkey for the Master's user.

gConnEvtKeyExchangeRequest_c The pairing procedure requires the SMP keys to be distributed to the peer. Data in gapConnectionEvent_t.eventData.keyExchangeRequestEvent.

gConnEvtKeysReceived_c SMP keys distributed by the peer during pairing have been received. Data in gapConnectionEvent_t.eventData.keysReceivedEvent.

gConnEvtLongTermKeyRequest_c The bonded peer Master has requested link encryption and the LTK must be provided. Slave should respond with Gap_ProvideLongTermKey. Data in gap↩ ConnectionEvent_t.eventData.longTermKeyRequestEvent.

gConnEvtEncryptionChanged_c Link's encryption state has changed, e.g., during pairing or after a reconnection with a bonded peer. Data in gapConnectionEvent_t.eventData.encryption↩ ChangedEvent.

gConnEvtPairingComplete_c Pairing procedure is complete, either successfully or with failure. Data in gapConnectionEvent_t.eventData.pairingCompleteEvent.

gConnEvtDisconnected_c A connection has been terminated. Data in gapConnectionEvent_t.↩ eventData.disconnectedEvent.

gConnEvtRssiRead_c RSSI for an active connection has been read. Data in gapConnectionEvent↩ _t.eventData.rssi_dBm.

gConnEvtTxPowerLevelRead_c TX power level for an active connection has been read. Data in gapConnectionEvent_t.eventData.txPowerLevel_dBm.

gConnEvtPowerReadFailure_c Power reading could not be performed. Data in gapConnection↩ Event_t.eventData.failReason.

gConnEvtParameterUpdateRequest_c A connection parameter update request has been received. Data in gapConnectionEvent_t.eventData.connectionUpdateRequest.

gConnEvtParameterUpdateComplete_c The connection has new parameters. Data in gap←ConnectionEvent_t.eventData.connectionUpdateComplete.

gConnEvtLeDataLengthChanged_c The new TX/RX Data Length parameters. Data in gap←ConnectionEvent_t.eventData.rssi_dBm.leDataLengthChanged.

gConnEvtLeScOobDataRequest_c Event sent to request LE SC OOB Data (r, Cr and Addr) received from a peer.

gConnEvtLeScDisplayNumericValue_c Event sent to display and confirm a Numeric Comparison Value when using the LE SC Numeric Comparison pairing method.

gConnEvtLeScKeypressNotification_c Remote Keypress Notification received during Passkey Entry Pairing Method.

gConnEvtChannelMapRead_c Channel Map was read for a connection. Data is contained in gap←ConnectionEvent_t.eventData.channelMap

gConnEvtChannelMapReadFailure_c Channel Map reading could not be performed. Data in gap←ConnectionEvent_t.eventData.failReason.

gConnEvtChanSelectionAlgorithm2_c LE Channel Selection Algorithm #2 is used on the data channel connection.

gConnEvtPairingNoLtk_c No LTK was found for the Master peer. Pairing shall be performed again.

3.5.15 enum gapCarSupport_t

Central Address Resolution characteristic state.

Enumerator

CAR_Unknown The Central Address Resolution characteristic was not read.

CAR_Unavailable The device tried to read the Central Address Resolution characteristic, but it's unavailable.

CAR_Unsupported The device has read the Central Address Resolution characteristic, and the it's value is FALSE.

CAR_Supported The device has read the Central Address Resolution characteristic, and the it's value is TRUE.

3.5.16 enum gapAppearance_t

Appearance characteristic enumeration, also used in advertising.

3.6 Function Documentation

3.6.1 bleResult_t Gap_RegisterDeviceSecurityRequirements (const gapDeviceSecurityRequirements_t ∗ *pSecurity*)

Registers the device security requirements. This function includes a master security for all services and, optionally, additional stronger security settings for services as required by the profile and/or application.

Parameters

in	*pSecurity*	A pointer to the application-allocated gapDeviceSecurity←Requirements_t structure.

Returns

gBleSuccess_c or error.

Remarks

pSecurity or any other contained security structure pointers that are NULL are ignored, i.e., defaulted to No Security (Security Mode 1 Level 1, No Authorization, Minimum encryption key size). This function executes synchronously.

3.6.2 bleResult_t Gap_SetAdvertisingParameters (const gapAdvertising←Parameters_t ∗ *pAdvertisingParameters*)

Sets up the Advertising Parameters.

Parameters

in	*pAdvertising←Parameters*	Pointer to gapAdvertisingParameters_t structure.

Returns

gBleSuccess_c or error.

Remarks

GAP Peripheral-only API function.

3.6.3 bleResult_t Gap_SetAdvertisingData (const gapAdvertisingData_t ∗ *pAdvertisingData,* const gapScanResponseData_t ∗ *pScanResponseData*)

Sets up the Advertising and Scan Response Data.

Parameters

in	*pAdvertising↩ Data*	Pointer to gapAdvertisingData_t structure or NULL.
in	*pScan↩ ResponseData*	Pointer to gapScanResponseData_t structure or NULL.

Returns

gBleSuccess_c or error.

Remarks

Any of the parameters may be NULL, in which case they are ignored. Therefore, this function can be used to set any of the parameters individually or both at once. The standard advertising packet payload is 37 bytes. Some of the payload may be occupied by the Advertiser Address which takes up 6 bytes and for some advertising PDU types also by the Initiator Address which takes another 6 bytes. This leaves 25-31 bytes to the application to include advertising structures (Length [1Byte], AD Type [1 Byte], AD Data[Length-1 Bytes])
GAP Peripheral-only API function.

3.6.4 bleResult_t Gap_StartAdvertising (gapAdvertisingCallback_t *advertisingCallback,* gapConnectionCallback_t *connectionCallback*)

Commands the controller to start advertising.

Parameters

in	*advertising↩ Callback*	Callback used by the application to receive advertising events. Can be NULL.
in	*connection↩ Callback*	Callback used by the application to receive connection events. Can be NULL.

Returns

gBleSuccess_c or error.

Remarks

The advertisingCallback confirms or denies whether the advertising has started. The connection↩ Callback is only used if a connection gets established during advertising.
GAP Peripheral-only API function.

3.6.5 bleResult_t Gap_StopAdvertising (void)

Commands the controller to stop advertising.

Returns

gBleSuccess_c or error.

Remarks

GAP Peripheral-only API function.

3.6.6 bleResult_t Gap_Authorize (deviceId_t *deviceId,* uint16_t *handle,* gattDbAccessType_t *access*)

Authorizes a peer for an attribute in the database.

Parameters

in	*deviceId*	The peer being authorized.
in	*handle*	The attribute handle.
in	*access*	The type of access granted (gAccessRead_c or gAccessWrite_c).

Returns

gBleSuccess_c or error.

Remarks

This function executes synchronously.
GATT Server-only API function.

3.6.7 bleResult_t Gap_SaveCccd (deviceId_t *deviceId,* uint16_t *handle,* gattCccdFlags_t *cccd*)

Save the CCCD value for a specific Client and CCCD handle.

Parameters

in	*deviceId*	The peer GATT Client.
in	*handle*	The handle of the CCCD as defined in the GATT Database.
in	*cccd*	The bit mask representing the CCCD value to be saved.

Returns

gBleSuccess_c or error.

Remarks

The GATT Server layer saves the CCCD value automatically when it is written by the Client. This API should only be used to save the CCCD in other situations, e.g., when for some reason the application decides to disable notifications/indications for a specific Client.
This function executes synchronously.
GATT Server-only API function.

3.6.8 bleResult_t Gap_CheckNotificationStatus (deviceId_t *deviceId,* uint16_t *handle,* bool_t ∗ *pOutIsActive*)

Retrieves the notification status for a given Client and a given CCCD handle.

Parameters

in	*deviceId*	The peer GATT Client.
in	*handle*	The handle of the CCCD.
out	*pOutIsActive*	The address to store the status into.

Returns

gBleSuccess_c or error.

Remarks

This function executes synchronously.
GATT Server-only API function.

3.6.9 bleResult_t Gap_CheckIndicationStatus (deviceId_t *deviceId,* uint16_t *handle,* bool_t ∗ *pOutIsActive*)

Retrieves the indication status for a given Client and a given CCCD handle.

Parameters

in	*deviceId*	The peer GATT Client.
in	*handle*	The handle of the CCCD.
out	*pOutIsActive*	The address to store the status into.

Returns

gBleSuccess_c or error.

Remarks

This function executes synchronously.
GATT Server-only API function.

3.6.10 bleResult_t Gap_GetBondedDevicesIdentityInformation (gapIdentity↩ Information_t ∗ *aOutIdentityAddresses,* uint8_t *maxDevices,* uint8_t ∗ *pOutActualCount*)

Retrieves a list of the identity information of bonded devices, if any.

Parameters

out	*aOutIdentity↩ Addresses*	Array of identities to be filled.
in	*maxDevices*	Maximum number of identities to be obtained.
out	*pOutActual↩ Count*	The actual number of identities written.

Returns

gBleSuccess_c or error.

Remarks

This API may be useful when creating a white list or a resolving list.
This function executes synchronously.

3.6.11 bleResult_t Gap_Pair (deviceId_t *deviceId,* const gapPairingParameters_t ∗ *pPairingParameters*)

Initiates pairing with a peer device.

Parameters

in	*deviceId*	The peer to pair with.
in	*pPairing←Parameters*	Pairing parameters as required by the SMP.

Returns

gBleSuccess_c or error.

Remarks

GAP Central-only API function.

3.6.12 bleResult_t Gap_SendSlaveSecurityRequest (deviceId_t *deviceId,* const gapPairingParameters_t ∗ *pPairingParameters*)

Informs the peer Master about the local security requirements.

Parameters

in	*deviceId*	The GAP peer to pair with.
in	*pPairing←Parameters*	Pairing parameters as required by the SMP.

Returns

gBleSuccess_c or error.

Remarks

The procedure has the same parameters as the pairing request, but, because it is initiated by the Slave, it has no pairing effect. It only informs the Master about the requirements.
GAP Peripheral-only API function.

3.6.13 bleResult_t Gap_EncryptLink (deviceId_t *deviceId*)

Encrypts the link with a bonded peer.

Parameters

in	*deviceId*	Device ID of the peer.

Returns

gBleSuccess_c or error.

Remarks

GAP Central-only API function.

3.6.14 bleResult_t Gap_AcceptPairingRequest (deviceId_t *deviceId,* const gapPairingParameters_t * *pPairingParameters*)

Accepts the pairing request from a peer.

Parameters

in	*deviceId*	The peer requesting authentication.
in	*pPairing↩ Parameters*	Pairing parameters as required by the SMP.

Returns

gBleSuccess_c or error.

Remarks

This should be called in response to a gPairingRequest_c event.
GAP Peripheral-only API function.

3.6.15 bleResult_t Gap_RejectPairing (deviceId_t *deviceId,* gapAuthentication↩ RejectReason_t *reason*)

Rejects the peer's authentication request.

Parameters

in	*deviceId*	The GAP peer who requested authentication.
in	*reason*	Reason why the current device rejects the authentication.

Returns

gBleSuccess_c or error.

3.6.16 bleResult_t Gap_EnterPasskey (deviceId_t *deviceId,* uint32_t *passkey*)

Enters the passkey requested by the peer during the pairing process.

Parameters

in	*deviceId*	The GAP peer that requested a passkey entry.
in	*passkey*	The peer's secret passkey.

Returns

gBleSuccess_c or error.

3.6.17 bleResult_t Gap_ProvideOob (deviceId_t *deviceId,* const uint8_t * *aOob*)

Provides the Out-Of-Band data for the SMP Pairing process.

Parameters

in	*deviceId*	The pairing device.
in	*aOob*	Pointer to OOB data (array of gcSmpOobSize_d size).

Returns

gBleSuccess_c or error.

3.6.18 bleResult_t Gap_RejectPasskeyRequest (deviceId_t *deviceId*)

Rejects the passkey request from a peer.

Parameters

in	*deviceId*	The GAP peer that requested a passkey entry.

Returns

gBleSuccess_c or error.

Remarks

GAP Central-only API function.

3.6.19 bleResult_t Gap_SendSmpKeys (deviceId_t *deviceId,* const gapSmpKeys_t ∗ *pKeys*)

Sends the SMP keys during the SMP Key Exchange procedure.

Parameters

in	*deviceId*	The GAP peer who initiated the procedure.
in	*pKeys*	The SMP keys of the local device.

Returns

gBleSuccess_c or error.

3.6.20 bleResult_t Gap_RejectKeyExchangeRequest (deviceId_t *deviceId*)

Rejects the Key Exchange procedure with a paired peer.

Parameters

in	*deviceId*	The GAP peer who requested the Key Exchange procedure.

Returns

gBleSuccess_c or error.

3.6.21 bleResult_t Gap_LeScRegeneratePublicKey (void)

Regenerates the private/public key pair used for LE Secure Connections pairing.

Returns

gBleSuccess_c or error.

Remarks

The application should listen for the gLeScPublicKeyRegenerated_c generic event.

3.6.22 bleResult_t Gap_LeScValidateNumericValue (deviceId_t *deviceId,* bool_t *valid*)

Validates the numeric value during the Numeric Comparison LE Secure Connections pairing.

Parameters

deviceId	Device ID of the peer.
valid	TRUE if user has indicated that numeric values are matched, FALSE otherwise.

Returns

gBleSuccess_c or error.

3.6.23 bleResult_t Gap_LeScGetLocalOobData (void)

Retrieves local OOB data used for LE Secure Connections pairing.

Returns

gBleSuccess_c or error.

Remarks

The application should listen for the gLeScLocalOobData_c generic event.

3.6.24 bleResult_t Gap_LeScSetPeerOobData (deviceId_t *deviceId,* const gapLeScOobData_t * *pPeerOobData*)

Sets peer OOB data used for LE Secure Connections pairing.

Parameters

deviceId	Device ID of the peer.
pPeerOobData	OOB data received from the peer.

Returns

gBleSuccess_c or error.

Remarks

This function should be called in response to the gConnEvtLeScOobDataRequest_c event.

3.6.25 bleResult_t Gap_LeScSendKeypressNotification (deviceId_t *deviceId,* gapKeypressNotification_t *keypressNotification*)

Sends a Keypress Notification to the peer.

Parameters

deviceId	Device ID of the peer.
keypress←Notification	Value of the Keypress Notification.

Returns

gBleSuccess_c or error.

Remarks

This function shall only be called during the passkey entry process and only if both peers support Keypress Notifications.

3.6.26 bleResult_t Gap_ProvideLongTermKey (deviceId_t *deviceId,* const uint8_t ∗ *aLtk,* uint8_t *ltkSize*)

Provides the Long Term Key (LTK) to the controller for encryption setup.

Parameters

in	*deviceId*	The GAP peer who requested encryption.
in	*aLtk*	The Long Term Key.
in	*ltkSize*	The Long Term Key size.

Returns

gBleSuccess_c or error.

Remarks

The application should provide the same LTK used during bonding with the respective peer.
GAP Peripheral-only API function.

3.6.27 bleResult_t Gap_DenyLongTermKey (deviceId_t *deviceId*)

Rejects an LTK request originating from the controller.

Parameters

in	*deviceId*	The GAP peer who requested encryption.

Returns

gBleSuccess_c or error.

Remarks

GAP Peripheral-only API function.

3.6.28 bleResult_t Gap_LoadEncryptionInformation (deviceId_t *deviceId,* uint8_t ∗ *aOutLtk,* uint8_t ∗ *pOutLtkSize*)

Loads the encryption key for a bonded device.

Parameters

in	*deviceId*	Device ID of the peer.
out	*aOutLtk*	Array of size gcMaxLtkSize_d to be filled with the LTK.
out	*pOutLtkSize*	The LTK size.

Returns

gBleSuccess_c or error.

Remarks

This function executes synchronously.

3.6.29 bleResult_t Gap_SetLocalPasskey (uint32_t *passkey*)

Sets the SMP passkey for this device.

Parameters

in	*passkey*	The SMP passkey.

Returns

gBleSuccess_c or error.

Remarks

This is the PIN that the peer's user must enter during pairing.
This function executes synchronously.
GAP Peripheral-only API function.

3.6.30 bleResult_t Gap_SetScanMode (gapScanMode_t *scanMode,* gapAuto↵ ConnectParams_t ∗ *pAutoConnectParams,* gapConnectionCallback_t *connCallback*)

Sets internal scan filters and actions.

Parameters

in	*scanMode*	The scan mode to be activated. Default is gDefaultScan_c.
in	*pAuto↩ Connect↩ Params*	Pointer to the gapAutoConnectParams_t structures if scanMode is set to gAutoConnect_c. The memory used must be persistent and should not change during the next scan periods or until another Gap_SetScanMode is called.
in	*connCallback*	Auto-Connect callback. Must be set if scanMode is set to gAuto↩ Connect_c.

Returns

gBleSuccess_c or error.

Remarks

This function can be called before Gap_StartScanning. If this function is never called, then the default value of gDefaultScan_c is considered and all scanned devices are reported to the application without any additional filtering or action.
This function executes synchronously.
GAP Central-only API function.

3.6.31 bleResult_t Gap_StartScanning (const gapScanningParameters_t ∗ *pScanningParameters,* gapScanningCallback_t *scanningCallback,* gapFilterDuplicates_t *enableFilterDuplicates,* uint16_t *duration,* uint16_t *period*)

Optionally sets the scanning parameters and begins scanning.

Parameters

in	*pScanning↩ Parameters*	The scanning parameters; may be NULL.
in	*scanning↩ Callback*	The scanning callback.
in	*enableFilter↩ Duplicates*	Enable or disable duplicate advertising report filtering
in	*duration*	Scan duration expressed in units of 10 ms. Set 0 for continuous scan until explicitly disabled. Used only for BLE5.0, otherwise ignored.

in	*period*	Time interval expressed in units of 1.28 seconds from when the Controller started its last Scan_Duration until it begins the subsequent Scan_Duration. Set 0 to disable periodic scanning. Used only for BL↩ E5.0, otherwise ignored.

Returns

gBleSuccess_c or error.

Remarks

Use this API to both set the scanning parameters and start scanning. If pScanningParameters is NULL, scanning is started with the existing settings.
GAP Central-only API function.

3.6.32 bleResult_t Gap_StopScanning (void)

Commands the controller to stop scanning.

Returns

gBleSuccess_c or error.

Remarks

GAP Central-only API function.

3.6.33 bleResult_t Gap_Connect (const gapConnectionRequestParameters_t ∗ *pParameters*, gapConnectionCallback_t *connCallback*)

Connects to a scanned device.

Parameters

in	*pParameters*	Create Connection command parameters.
in	*connCallback*	Callback used to receive connection events.

Returns

gBleSuccess_c or error.

Remarks

GAP Central-only API function.

3.6.34 bleResult_t Gap_Disconnect (deviceId_t *deviceId*)

Initiates disconnection from a connected peer device.

Parameters

in	*deviceId*	The connected peer to disconnect from.

Returns

gBleSuccess_c or error.

3.6.35 bleResult_t Gap_SaveCustomPeerInformation (deviceId_t *deviceId,* const uint8_t ∗ *aInfo,* uint16_t *offset,* uint16_t *infoSize*)

Saves custom peer information in raw data format.

Parameters

in	*deviceId*	Device ID of the GAP peer.
in	*aInfo*	Pointer to the beginning of the data.
in	*offset*	Offset from the beginning of the reserved memory area.
in	*infoSize*	Data size (maximum equal to gcReservedFlashSizeForCustom←Information_d).

Returns

gBleSuccess_c or error.

Remarks

This function can be called by the application to save custom information about the peer device, e.g., Service Discovery data (to avoid doing it again on reconnection).
This function executes synchronously.

3.6.36 bleResult_t Gap_LoadCustomPeerInformation (deviceId_t *deviceId,* uint8_t ∗ *aOutInfo,* uint16_t *offset,* uint16_t *infoSize*)

Loads the custom peer information in raw data format.

Parameters

in	*deviceId*	Device ID of the GAP peer.

out	*aOutInfo*	Pointer to the beginning of the allocated memory.
in	*offset*	Offset from the beginning of the reserved memory area.
in	*infoSize*	Data size (maximum equal to gcReservedFlashSizeForCustom←Information_d).

Returns

gBleSuccess_c or error.

Remarks

This function can be called by the application to load custom information about the peer device, e.g., Service Discovery data (to avoid doing it again on reconnection).
This function executes synchronously.

3.6.37 bleResult_t Gap_CheckIfBonded (deviceId_t *deviceId,* bool_t ∗ *pOutIsBonded,* uint8_t ∗ *pOutNvmIndex*)

Returns whether or not a connected peer device is bonded and the NVM index.

Parameters

in	*deviceId*	Device ID of the GAP peer.
out	*pOutIsBonded*	Boolean to be filled with the bonded flag.
out	*pOutNvmIndex*	If bonded, to be filled optionally with the NVM index.

Returns

gBleSuccess_c or error.

Remarks

This function executes synchronously.

3.6.38 bleResult_t Gap_ReadWhiteListSize (void)

Retrieves the size of the White List.

Returns

gBleSuccess_c or error.

Remarks

Response is received in the gWhiteListSizeReady_c generic event.

3.6.39 bleResult_t Gap_ClearWhiteList (void)

Removes all addresses from the White List, if any.

Returns

gBleSuccess_c or error.

Remarks

Confirmation is received in the gWhiteListCleared_c generic event.

3.6.40 bleResult_t Gap_AddDeviceToWhiteList (bleAddressType_t *addressType,* const bleDeviceAddress_t *address*)

Adds a device address to the White List.

Parameters

in	*address*	The address of the white-listed device.
in	*addressType*	The device address type (public or random).

Returns

gBleSuccess_c or error.

3.6.41 bleResult_t Gap_RemoveDeviceFromWhiteList (bleAddressType_t *addressType,* const bleDeviceAddress_t *address*)

Removes a device address from the White List.

Parameters

in	*address*	The address of the white-listed device.
in	*addressType*	The device address type (public or random).

Returns

gBleSuccess_c or error.

3.6.42 bleResult_t Gap_ReadPublicDeviceAddress (void)

Reads the device's public address from the controller.

Returns

gBleSuccess_c or error.

Remarks

The application should listen for the gPublicAddressRead_c generic event.

3.6.43 bleResult_t Gap_CreateRandomDeviceAddress (const uint8_t * *aIrk*, const uint8_t * *aRandomPart*)

Requests the controller to create a random address.

Parameters

in	*aIrk*	The Identity Resolving Key to be used for a private resolvable address or NULL for a private non-resolvable address (fully random).
in	*aRandomPart*	If aIrk is not NULL, this is a 3-byte array containing the Random Part of a Private Resolvable Address, in LSB to MSB order; the most significant two bits of the most significant byte (aRandomPart[3] & 0xC0) are ignored. This may be NULL, in which case the Random Part is randomly generated internally.

Returns

gBleSuccess_c or error.

Remarks

The application should listen for the gRandomAddressReady_c generic event. Note that this does not set the random address in the Controller. To set the random address, call Gap_SetRandomAddress() with the generated address contained in the event data.

3.6.44 bleResult_t Gap_SaveDeviceName (deviceId_t *deviceId*, const uchar_t * *aName*, uint8_t *cNameSize*)

Store the name of a bonded device.

Parameters

in	*deviceId*	Device ID for the active peer which name is saved.
in	*aName*	Array of characters holding the name.
in	*cNameSize*	Number of characters to be saved.

Returns

gBleSuccess_c or error.

Remarks

This function copies cNameSize characters from the aName array and adds the NULL character to terminate the string.
This function executes synchronously.

3.6.45 bleResult_t Gap_GetBondedDevicesCount (uint8_t ∗ *pOutBondedDevicesCount*)

Retrieves the number of bonded devices.

Parameters

out	*pOutBonded↩ DevicesCount*	Pointer to integer to be written.

Returns

gBleSuccess_c or error.

Remarks

This function executes synchronously.

3.6.46 bleResult_t Gap_GetBondedDeviceName (uint8_t *nvmIndex,* uchar_t ∗ *aOutName,* uint8_t *maxNameSize*)

Retrieves the name of a bonded device.

Parameters

`in`	*nvmIndex*	Index of the device in NVM bonding area.
`out`	*aOutName*	Destination array to copy the name into.
`in`	*maxNameSize*	Maximum number of characters to be copied, including the terminating NULL character.

Returns

gBleSuccess_c or error.

Remarks

nvmIndex is an integer ranging from 0 to N-1, where N is the number of bonded devices and can be obtained by calling Gap_GetBondedDevicesCount(&N).
This function executes synchronously.

3.6.47 bleResult_t Gap_RemoveBond (uint8_t *nvmIndex*)

Removes the bond with a device.

Parameters

`in`	*nvmIndex*	Index of the device in the NVM bonding area.

Returns

gBleSuccess_c or error.

Remarks

This API requires that there are no active connections at call time. nvmIndex is an integer ranging from 0 to N-1, where N is the number of bonded devices and can be obtained by calling Gap_Get↩ BondedDevicesCount(&N).
This function executes synchronously.

3.6.48 bleResult_t Gap_RemoveAllBonds (void)

Removes all bonds with other devices.

Returns

gBleSuccess_c or error.

Remarks

This API requires that there are no active connections at call time.
This function executes synchronously.

3.6.49 bleResult_t Gap_ReadRadioPowerLevel (gapRadioPowerLevelReadType_t *txReadType,* deviceId_t *deviceId*)

Reads the power level of the controller's radio.

Returns

gBleSuccess_c or error.

Remarks

The response is contained in the gConnEvtTxPowerLevelRead_c connection event when reading connection TX power level, the gAdvTxPowerLevelRead_c generic event when reading the advertising TX power level, or the gConnEvtRssiRead_c connection event when reading the RSSI.

3.6.50 bleResult_t Gap_SetTxPowerLevel (uint8_t *powerLevel,* bleTransmitPowerChannelType_t *channelType*)

Sets the Tx power level on the controller's radio.

Parameters

`in`	*powerLevel*	Power level as specified in the controller interface.
`in`	*channelType*	The advertising or connection channel type.

Returns

gBleSuccess_c or error.

Remarks

The application should listen for the gTxPowerLevelSetComplete_c generic event.
This function executes synchronously.

3.6.51 bleResult_t Gap_VerifyPrivateResolvableAddress (uint8_t *nvmIndex,* const bleDeviceAddress_t *aAddress*)

Verifies a Private Resolvable Address with a bonded device's IRK.

Parameters

in	*nvmIndex*	Index of the device in NVM bonding area whose IRK must be checked.
in	*aAddress*	The Private Resolvable Address to be verified.

Returns

gBleSuccess_c or error.

Remarks

nvmIndex is an integer ranging from 0 to N-1, where N is the number of bonded devices and can be obtained by calling Gap_GetBondedDevicesCount(&N); the application should listen to the g↩ PrivateResolvableAddressVerified_c event.

3.6.52 bleResult_t Gap_SetRandomAddress (const bleDeviceAddress_t *aAddress*)

Sets a random address into the Controller.

Parameters

in	*aAddress*	The Private Resolvable, Private Non-Resolvable, or Static Random Address.

Returns

gBleSuccess_c or error.

Remarks

The application should listen for the gRandomAddressSet_c generic event.

3.6.53 bleResult_t Gap_SetDefaultPairingParameters (const gapPairingParameters_t ∗ *pPairingParameters*)

Sets the default pairing parameters to be used by automatic pairing procedures.

Parameters

in	*pPairing↩ Parameters*	Pairing parameters as required by the SMP or NULL.

Returns

gBleSuccess_c or error.

Remarks

When these parameters are set, the Security Manager automatically responds to a Pairing Request or a Slave Security Request using these parameters. If NULL is provided, it returns to the default state where all security requests are sent to the application.
This function executes synchronously.

3.6.54 bleResult_t Gap_UpdateConnectionParameters (deviceId_t *deviceId,* uint16_t *intervalMin,* uint16_t *intervalMax,* uint16_t *slaveLatency,* uint16_t *timeoutMultiplier,* uint16_t *minCeLength,* uint16_t *maxCeLength*)

Request a set of new connection parameters

Parameters

in	*deviceId*	The DeviceID for which the command is intended
in	*intervalMin*	The minimum value for the connection event interval
in	*intervalMax*	The maximum value for the connection event interval
in	*slaveLatency*	The slave latency parameter
in	*timeout↩ Multiplier*	The connection timeout parameter
in	*minCeLength*	The minimum value for the connection event length
in	*maxCeLength*	The maximum value for the connection event length

Returns

gBleSuccess_c or error.

Precondition

A connection must be in place

3.6.55 bleResult_t Gap_EnableUpdateConnectionParameters (deviceId_t *deviceId,* bool_t *enable*)

Update the connection parameters

Parameters

`in`	*deviceId*	The DeviceID for which the command is intended
`in`	*enable*	Allow/disallow the parameters update

Returns

Result of the operation

Precondition

A connection must be in place

Remarks

The LE master Host may accept the requested parameters or reject the request

3.6.56 bleResult_t Gap_UpdateLeDataLength (deviceId_t *deviceId,* uint16_t *txOctets,* uint16_t *txTime*)

Update the Tx Data Length

Parameters

`in`	*deviceId*	The DeviceID for which the command is intended
`in`	*txOctets*	Maximum transmission number of payload octets
`in`	*txTime*	Maximum transmission time

Returns

Result of the operation

Precondition

A connection must be in place

Remarks

The response is contained in the gConnEvtLeDataLengthUpdated_c connection event.

3.6.57 bleResult_t Gap_EnableHostPrivacy (bool_t *enable,* const uint8_t ∗ *aIrk*)

Enables or disables Host Privacy (automatic regeneration of a Private Address).

Parameters

enable	TRUE to enable, FALSE to disable.
aIrk	Local IRK to be used for Resolvable Private Address generation or NULL for Non-↩ Resolvable Private Address generation. Ignored if enable is FALSE.

Returns

gBleSuccess_c or error.

Remarks

The application should listen for the gHostPrivacyStateChanged_c generic event.

3.6.58 bleResult_t Gap_EnableControllerPrivacy (bool_t *enable,* const uint8_t ∗ *aOwnIrk,* uint8_t *peerIdCount,* const gapIdentityInformation_t ∗ *aPeerIdentities*)

Enables or disables Controller Privacy (Enhanced Privacy feature).

Parameters

enable	TRUE to enable, FALSE to disable.
aOwnIrk	Local IRK. Ignored if enable is FALSE, otherwise shall not be NULL.
peerIdCount	Size of aPeerIdentities array. Shall not be zero or greater than gcGapController↩ ResolvingListSize_c. Ignored if enable is FALSE.
aPeerIdentities	Array of peer identity addresses and IRKs. Ignored if enable is FALSE, otherwise shall not be NULL.

Returns

gBleSuccess_c or error.

Remarks

The application should listen for the gControllerPrivacyStateChanged_c generic event.

3.6.59 bleResult_t Gap_SetPrivacyMode (uint8_t *nvmIndex,* blePrivacyMode_t *privacyMode*)

Sets the privacy mode to an existing bond.

Parameters

in	*nvmIndex*	Index of the device in the NVM bonding area.
in	*privacyMode*	Controller privacy mode: Network or Device

Returns

gBleSuccess_c or error.

Remarks

The change has no effect (other than the change in NVM) unless controller privacy is enabled for the bonded identities.

3.6.60 bleResult_t Gap_ControllerTest (gapControllerTestCmd_t *testCmd*, uint8_t *radioChannel*, uint8_t *txDataLength*, gapControllerTestTxType_t *txPayloadType*)

Commands a Controller Test procedure.

Parameters

testCmd	Command type - "start TX test", "start RX test" or "end test".
radioChannel	Radio channel index. Valid range: 0-39. Frequency will be F[MHz] = 2402 + 2 $*$ index. Effective range: 2402-2480 MHz. Ignored if command is "end test".
txDataLength	Size of packet payload for TX tests. Ignored if command is "start RX test" or "end test".
txPayloadType	Type of packet payload for TX tests. Ignored if command is "start RX test" or "end test".

Returns

gBleSuccess_c or error.

Remarks

The application should listen for the gControllerTestEvent_c generic event.
This API function is available only in the full-featured host library.

3.6.61 bleResult_t Gap_LeReadPhy (deviceId_t *deviceId*)

Read the Tx and Rx Phy on the connection with a device

Parameters

deviceId	Device ID of the peer.

Returns

gBleSuccess_c or error.

Remarks

The application should listen for the gLePhyEvent_c generic event. This API is available only in the Bluetooth 5.0 Host Stack.

3.6.62 bleResult_t Gap_LeSetPhy (bool_t *defaultMode,* deviceId_t *deviceId,* uint8_t *allPhys,* uint8_t *txPhys,* uint8_t *rxPhys,* uint16_t *phyOptions*)

Set the Tx and Rx Phy preferences on the connection with a device or all subsequent connections

Parameters

defaultMode	Use the provided values for all subsequent connections
deviceId	Device ID of the peer Ignored if defaultMode is TRUE.
allPhys	Host preferences on Tx and Rx Phy, as defined by gapLeAllPhyFlags_t
txPhys	Host preferences on Tx Phy, as defined by gapLePhyFlags_t, ignored for gLeTxPhy↩ NoPreference_c
rxPhys	Host preferences on Rx Phy, as defined by gapLePhyFlags_t, ignored for gLeRx↩ PhyNoPreference_c
phyOptions	Host preferences on Coded Phy, as defined by gapLePhyOptionsFlags_t Ignored if defaultMode is TRUE.

Returns

gBleSuccess_c or error.

Remarks

The application should listen for the gLePhyEvent_c generic event. This API is available only in the Bluetooth 5.0 Host Stack.

3.6.63 bleResult_t Gap_ControllerEnhancedNotification (uint16_t *eventType,* deviceId_t *deviceId*)

Configure enhanced notifications on advertising, scanning and connection events on the controller.

Parameters

in	*eventType*	Event type selection as specified by bleNotificationEventType_t.
in	*deviceId*	Device ID of the peer, used only for connection events.

Returns

gBleSuccess_c or error.

Remarks

The application should listen for the gControllerNotificationEvent_c generic event.
This function executes synchronously.

3.6.64 bleResult_t Gap_LoadKeys (uint8_t *nvmIndex,* gapSmpKeys_t ∗ *pOutKeys,* gapSmpKeyFlags_t ∗ *pOutKeyFlags,* bool_t ∗ *pOutLeSc,* bool_t ∗ *pOutAuth*)

Retrieves the keys from an existing bond with a device.

Parameters

in	*nvmIndex*	Index of the device in the NVM bonding area.
out	*pOutKeys*	Pointer to fill the keys distributed during pairing.
out	*pOutKeyFlags*	Pointer to indicate which keys were distributed during pairing.
out	*pOutLeSc*	Pointer to mark if LE Secure Connections was used during pairing.
out	*pOutAuth*	Pointer to mark if the device was authenticated for MITM during pairing.

Returns

gBleSuccess_c or error.

Remarks

This API requires that the aAddress in the pOutKeys shall not be NULL.
The application will check pOutKeyFlags to see which information is valid in pOutKeys.
This function executes synchronously.

3.6.65 bleResult_t Gap_SaveKeys (uint8_t *nvmIndex,* const gapSmpKeys_t ∗ *pKeys,* bool_t *leSc,* bool_t *auth*)

Saves the keys to a new or existing bond based on OOB information.

Parameters

in	*nvmIndex*	Index of the device in the NVM bonding area.
in	*pKeys*	Pointer to the keys distributed during pairing.
in	*leSc*	Indicates if LE Secure Connections was used during pairing.
in	*auth*	Indicates if the device was authenticated for MITM during pairing.

Returns

gBleSuccess_c or error.

Remarks

This API requires that the aAddress in the pKeys shall not be NULL.
If any of the keys are passed as NULL, they will not be saved.
The application listen for gBondCreatedEvent_c to confirm the bond was created.

3.6.66 bleResult_t Gap_SetChannelMap (const bleChannelMap_t *channelMap*)

Set the channel map in the Controller and trigger a LL channel map update.

Parameters

in	*channelMap*	Array with the channels using 0 for bad channels and 1 for unknown.

Returns

gBleSuccess_c or error.

Remarks

The application should listen for the gChannelMapSet_c generic event.
This function executes synchronously.
GAP Central-only API function.

3.6.67 bleResult_t Gap_ReadChannelMap (deviceId_t *deviceId*)

Reads the channel map of a connection.

Parameters

in	*deviceId*	Device ID of the peer.

Returns

gBleSuccess_c or error.

Remarks

The application should listen for the gConnEvtChannelMapRead_c connection event.
This function executes synchronously.

3.6.68 bleResult_t Gap_SetExtAdvertisingParameters (gapExtAdvertising←Parameters_t ∗ *pAdvertisingParameters*)

Sets up the Extended Advertising Parameters.

Parameters

in	*pAdvertising←Parameters*	Pointer to gapExtAdvertisingParameters_t structure.

Returns

gBleSuccess_c or error.

Remarks

GAP Peripheral-only API function.

3.6.69 bleResult_t Gap_SetExtAdvertisingData (uint8_t *handle,* gapAdvertising←Data_t ∗ *pAdvertisingData,* gapScanResponseData_t ∗ *pScanResponseData*)

Sets up the Extended Advertising and Extended Scan Response Data.

Parameters

in	*handle*	The ID of the advertising set
in	*pAdvertising↩ Data*	Pointer to gapAdvertisingData_t structure or NULL.
in	*pScan↩ ResponseData*	Pointer to gapScanResponseData_t structure or NULL.

Returns

gBleSuccess_c or error.

Remarks

Any of the parameters may be NULL, in which case they are ignored. Therefore, this function can be used to set any of the parameters individually or both at once.
GAP Peripheral-only API function.

3.6.70 bleResult_t Gap_StartExtAdvertising (gapAdvertisingCallback_t *advertisingCallback,* gapConnectionCallback_t *connectionCallback,* uint8_t *handle,* uint16_t *duration,* uint8_t *maxExtAdvEvents*)

Commands the controller to start the extended advertising.

Parameters

in	*advertising↩ Callback*	Callback used by the application to receive advertising events. Can be NULL.
in	*connection↩ Callback*	Callback used by the application to receive connection events. Can be NULL.
in	*handle*	The ID of the advertising set
in	*duration*	The duration of the advertising
in	*maxExtAdv↩ Events*	The maximum number of advertising events

Returns

gBleSuccess_c or error.

Remarks

The advertisingCallback confirms or denies whether the advertising has started. The connection↩ Callback is only used if a connection gets established during advertising.
GAP Peripheral-only API function.

3.6.71 bleResult_t Gap_StopExtAdvertising (uint8_t *handle*)

Commands the controller to stop extended advertising for set ID.

Parameters

in	*handle*	The ID of the advertising set

Returns

gBleSuccess_c or error.

Remarks

GAP Peripheral-only API function.

3.6.72 bleResult_t Gap_RemoveAdvSet (uint8_t *handle*)

Commands the controller to remove the specified advertising set and all it's data.

Parameters

in	*handle*	The ID of the advertising set

Returns

gBleSuccess_c or error.

Remarks

GAP Peripheral-only API function.

3.6.73 bleResult_t Gap_SetPeriodicAdvParameters (gapPeriodicAdvParameters_t ∗ *pAdvertisingParameters*)

Sets up the Periodic Advertising Parameters.

Parameters

in	*pAdvertising←Parameters*	Pointer to gapPeriodicAdvParameters_t structure.

Returns

gBleSuccess_c or error.

Remarks

GAP Peripheral-only API function.

3.6.74 bleResult_t Gap_SetPeriodicAdvertisingData (uint8_t *handle*, gapAdvertisingData_t ∗ *pAdvertisingData*)

Sets up the Periodic Advertising Data.

Parameters

in	*handle*	The ID of the periodic advertising set
in	*pAdvertising←Data*	Pointer to gapAdvertisingData_t structure.

Returns

gBleSuccess_c or error.

Remarks

GAP Peripheral-only API function.

3.6.75 bleResult_t Gap_StartPeriodicAdvertising (uint8_t *handle*)

Commands the controller to start periodic advertising for set ID.

Parameters

in	*handle*	The ID of the periodic advertising set

Returns

gBleSuccess_c or error.

Remarks

GAP Peripheral-only API function.

3.6.76 bleResult_t Gap_StopPeriodicAdvertising (uint8_t *handle*)

Commands the controller to stop periodic advertising for set ID.

Parameters

in	*handle*	The ID of the periodic advertising set

Returns

gBleSuccess_c or error.

Remarks

GAP Peripheral-only API function.

3.6.77 bleResult_t Gap_UpdatePeriodicAdvList (gapPeriodicAdvListOperation_t *operation,* bleAddressType_t *addrType,* uint8_t ∗ *pAddr,* uint8_t *SID*)

Manage the periodic advertising list.

Parameters

in	*operation*	The list operation: add/remove a device, or clear all.
in	*addrType*	The address type of the periodic advertiser.
in	*pAddr*	Pointer to the advertiser's address.
in	*SID*	The ID of the advertising set.

Returns

gBleSuccess_c or error.

Remarks

GAP Central-only API function.

3.6.78 bleResult_t Gap_PeriodicAdvCreateSync (gapPeriodicAdvSyncReq_t * *pReq*)

Start tracking periodic advertisings. Scanning is required to be ON for this request to be processed, so the scanning callback will receive the periodic advertising events.

Parameters

in	*pReq*	Pointer to the Sync Request parameters.

Returns

gBleSuccess_c or error.

Remarks

GAP Central-only API function.

3.6.79 bleResult_t Gap_PeriodicAdvTerminateSync (uint16_t *syncHandle*)

Stop tracking periodic advertisings.

Parameters

in	*syncHandle*	Used to identify the periodic advertiser

Returns

gBleSuccess_c or error.

Remarks

GAP Central-only API function.

3.6.80 bleResult_t Gap_ResumeLeScStateMachine (computeDhKeyParam_t ∗ *pData*)

Resume the pairing process. At this point the ecdh key must be computed. This function should be called only for secured LE connections. In any other cases the user should make his own code for handling the case when the ECDH computation is completed.

Parameters

in	*pData*	Pointer to the data used to resume the host state machine. The data is allocated by the stack when it requested an ECDH multiplication. It is also freed by the stack at the end of the multiplication.

Returns

status of the procedure.

Chapter 4
GATT - Generic Attribute Profile Interface

4.1 Overview

Files

- file att_errors.h
- file gatt_types.h
- file gatt_interface.h

Data Structures

- struct gattAttribute_t
- struct gattCharacteristic_t
- struct gattService_t
- struct gattDbCharPresFormat_t
- struct gattHandleRange_t

Macros

- #define gCccdEmpty_c
- #define gCccdNotification_c
- #define gCccdIndication_c

Typedefs

- typedef uint8_t gattCccdFlags_t

Enumerations

- enum attErrorCode_t {
  gAttErrCodeNoError_c,
  gAttErrCodeInvalidHandle_c,
  gAttErrCodeReadNotPermitted_c,
  gAttErrCodeWriteNotPermitted_c,
  gAttErrCodeInvalidPdu_c,
  gAttErrCodeInsufficientAuthentication_c,
  gAttErrCodeRequestNotSupported_c,
  gAttErrCodeInvalidOffset_c,
  gAttErrCodeInsufficientAuthorization_c,
  gAttErrCodePrepareQueueFull_c,
  gAttErrCodeAttributeNotFound_c,
  gAttErrCodeAttributeNotLong_c,
  gAttErrCodeInsufficientEncryptionKeySize_c,
  gAttErrCodeInvalidAttributeValueLength_c,
  gAttErrCodeUnlikelyError_c,
  gAttErrCodeInsufficientEncryption_c,
  gAttErrCodeUnsupportedGroupType_c,
  gAttErrCodeInsufficientResources_c,
  gAttErrCodeWriteRequestRejected_c,
  gAttErrCodeCccdImproperlyConfigured_c,
  gAttErrCodeProcedureAlreadyInProgress_c,
  gAttErrCodeOutOfRange_c }

Functions

- bleResult_t Gatt_Init (void)
- bleResult_t Gatt_GetMtu (deviceId_t deviceId, uint16_t ∗pOutMtu)

4.2 Data Structure Documentation

4.2.1 struct gattAttribute_t

GATT Attribute structure definition.

Data Fields

uint16_t	handle	Attribute handle.
bleUuidType↩_t	uuidType	Type of the UUID.

bleUuid_t	uuid	The attribute's UUID.
uint16_t	valueLength	Length of the attribute value array.
uint16_t	maxValue↩ Length	Maximum length of the attribute value array; if this is set to 0, then the attribute's length is fixed and cannot be changed.
uint8_t *	paValue	Attribute value array.

4.2.2 struct gattCharacteristic_t

GATT Characteristic structure definition.

Data Fields

gatt↩ Characteristic↩ PropertiesBit↩ Fields_t	properties	Characteristic Properties as defined by GATT.
gattAttribute↩ _t	value	Characteristic Value attribute.
uint8_t	cNum↩ Descriptors	Size of the Characteristic Descriptors array.
gattAttribute↩ _t *	aDescriptors	Characteristic Descriptors array.

4.2.3 struct gattService_t

GATT Service structure definition.

Data Fields

uint16_t	startHandle	The handle of the Service Declaration attribute.
uint16_t	endHandle	The last handle belonging to this Service (followed by another Service declaration of the end of the database).
bleUuidType↩_t	uuidType	Service UUID type.
bleUuid_t	uuid	Service UUID.
uint8_t	cNum↩Characteristics	Size of the Characteristic array.
gatt↩Characteristic↩_t *	a↩Characteristics	Characteristic array.
uint8_t	cNum↩Included↩Services	Size of the Included Services array.
struct gattService_tag *	aIncluded↩Services	Included Services array.

4.2.4 struct gattDbCharPresFormat_t

Characteristic Presentation Format Descriptor structure.

Data Fields

uint8_t	format	
int8_t	exponent	
uint16_t	unitUuid16	
uint8_t	ns	
uint16_t	description	

4.2.5 struct gattHandleRange_t

GATT Handle Range structure definition.

Data Fields

uint16_t	startHandle	Start Handle.
uint16_t	endHandle	End Handle - shall be greater than or equal to Start Handle.

4.3 Macro Definition Documentation

4.3.1 #define gCccdEmpty_c

Nothing is enabled.

4.3.2 #define gCccdNotification_c

Enables notifications.

4.3.3 #define gCccdIndication_c

Enabled indications.

4.4 Typedef Documentation

4.4.1 typedef uint8_t gattCccdFlags_t

Flags for the value of the Client Characteristic Configuration Descriptor.

4.5 Enumeration Type Documentation

4.5.1 enum attErrorCode_t

ATT error codes.

4.6 Function Documentation

4.6.1 bleResult_t Gatt_Init (void)

Initializes the GATT module.

Remarks

> If the GAP module is present, this function is called internally by Ble_HostInitialize(). Otherwise, the application must call this function once at device start-up.
> This function executes synchronously.

4.6.2 bleResult_t Gatt_GetMtu (deviceId_t *deviceId,* uint16_t ∗ *pOutMtu*)

Retrieves the MTU used with a given connected device.

Parameters

in	*deviceId*	The device ID of the connected peer.
out	*pOutMtu*	Pointer to integer to be written.

Returns

gBleSuccess_c or error.

Remarks

This function executes synchronously.

Chapter 5
GATT - Client APIs

5.1 Overview

Files

- file gatt_client_interface.h

Macros

- #define GattClient_SimpleCharacteristicWrite(deviceId, pChar, valueLength, aValue)
- #define GattClient_CharacteristicWriteWithoutResponse(deviceId, pChar, valueLength, aValue)
- #define GattClient_CharacteristicSignedWrite(deviceId, pChar, valueLength, aValue, aCsrk)

Typedefs

- typedef void(∗ gattClientProcedureCallback_t) (deviceId_t deviceId, gattProcedureType_↩ t procedureType, gattProcedureResult_t procedureResult, bleResult_t error)
- typedef void(∗ gattClientNotificationCallback_t) (deviceId_t deviceId, uint16_t characteristic↩ ValueHandle, uint8_t ∗aValue, uint16_t valueLength)
- typedef gattClientNotificationCallback_t gattClientIndicationCallback_t

Enumerations

- enum gattProcedureType_t {
  gGattProcExchangeMtu_c,
  gGattProcDiscoverAllPrimaryServices_c,
  gGattProcDiscoverPrimaryServicesByUuid_c,
  gGattProcFindIncludedServices_c,
  gGattProcDiscoverAllCharacteristics_c,
  gGattProcDiscoverCharacteristicByUuid_c,
  gGattProcDiscoverAllCharacteristicDescriptors_c,
  gGattProcReadCharacteristicValue_c,
  gGattProcReadUsingCharacteristicUuid_c,
  gGattProcReadMultipleCharacteristicValues_c,
  gGattProcWriteCharacteristicValue_c,
  gGattProcReadCharacteristicDescriptor_c,
  gGattProcWriteCharacteristicDescriptor_c }
- enum gattProcedureResult_t {
  gGattProcSuccess_c,
  gGattProcError_c }

Functions

- bleResult_t GattClient_Init (void)
- bleResult_t GattClient_ResetProcedure (void)
- bleResult_t GattClient_RegisterProcedureCallback (gattClientProcedureCallback_t callback)
- bleResult_t GattClient_RegisterNotificationCallback (gattClientNotificationCallback_t callback)
- bleResult_t GattClient_RegisterIndicationCallback (gattClientIndicationCallback_t callback)
- bleResult_t GattClient_ExchangeMtu (deviceId_t deviceId, uint16_t mtu)
- bleResult_t GattClient_DiscoverAllPrimaryServices (deviceId_t deviceId, gattService_t ∗aOut↩ PrimaryServices, uint8_t maxServiceCount, uint8_t ∗pOutDiscoveredCount)
- bleResult_t GattClient_DiscoverPrimaryServicesByUuid (deviceId_t deviceId, bleUuidType_↩ t uuidType, const bleUuid_t ∗pUuid, gattService_t ∗aOutPrimaryServices, uint8_t maxService↩ Count, uint8_t ∗pOutDiscoveredCount)
- bleResult_t GattClient_FindIncludedServices (deviceId_t deviceId, gattService_t ∗pIoService, uint8_t maxServiceCount)
- bleResult_t GattClient_DiscoverAllCharacteristicsOfService (deviceId_t deviceId, gattService_↩ t ∗pIoService, uint8_t maxCharacteristicCount)
- bleResult_t GattClient_DiscoverCharacteristicOfServiceByUuid (deviceId_t deviceId, bleUuid↩ Type_t uuidType, const bleUuid_t ∗pUuid, const gattService_t ∗pService, gattCharacteristic_t ∗a↩ OutCharacteristics, uint8_t maxCharacteristicCount, uint8_t ∗pOutDiscoveredCount)
- bleResult_t GattClient_DiscoverAllCharacteristicDescriptors (deviceId_t deviceId, gattCharacteristic↩ _t ∗pIoCharacteristic, uint16_t endingHandle, uint8_t maxDescriptorCount)
- bleResult_t GattClient_ReadCharacteristicValue (deviceId_t deviceId, gattCharacteristic_t ∗pIo↩ Characteristic, uint16_t maxReadBytes)
- bleResult_t GattClient_ReadUsingCharacteristicUuid (deviceId_t deviceId, bleUuidType_t uuid↩ Type, const bleUuid_t ∗pUuid, const gattHandleRange_t ∗pHandleRange, uint8_t ∗aOutBuffer, uint16_t maxReadBytes, uint16_t ∗pOutActualReadBytes)
- bleResult_t GattClient_ReadMultipleCharacteristicValues (deviceId_t deviceId, uint8_t cNum↩ Characteristics, gattCharacteristic_t ∗aIoCharacteristics)
- bleResult_t GattClient_WriteCharacteristicValue (deviceId_t deviceId, const gattCharacteristic_↩ t ∗pCharacteristic, uint16_t valueLength, const uint8_t ∗aValue, bool_t withoutResponse, bool_t signedWrite, bool_t doReliableLongCharWrites, const uint8_t ∗aCsrk)
- bleResult_t GattClient_ReadCharacteristicDescriptor (deviceId_t deviceId, gattAttribute_t ∗pIo↩ Descriptor, uint16_t maxReadBytes)
- bleResult_t GattClient_WriteCharacteristicDescriptor (deviceId_t deviceId, const gattAttribute_↩ t ∗pDescriptor, uint16_t valueLength, const uint8_t ∗aValue)

5.2 Macro Definition Documentation

5.2.1 #define GattClient_SimpleCharacteristicWrite(*deviceId*, *pChar*, *valueLength*, *aValue*)

Executes the basic Characteristic Write operation (with server confirmation).

Parameters

in	*deviceId*	Device ID of the connected GATT Server.
in	*pChar*	Pointer to the Characteristic being written.
in	*valueLength*	Size in bytes of the value to be written.
in	*aValue*	Array of bytes to be written.

Returns

gBleSuccess_c or error.

5.2.2 #define GattClient_CharacteristicWriteWithoutResponse(*deviceId, pChar, valueLength, aValue*)

Executes the Characteristic Write Without Response operation.

Parameters

in	*deviceId*	Device ID of the connected GATT Server.
in	*pChar*	Pointer to the Characteristic being written.
in	*valueLength*	Size in bytes of the value to be written.
in	*aValue*	Array of bytes to be written.

Returns

gBleSuccess_c or error.

5.2.3 #define GattClient_CharacteristicSignedWrite(*deviceId, pChar, valueLength, aValue, aCsrk*)

Executes the Characteristic Signed Write Without Response operation.

Parameters

in	*deviceId*	Device ID of the connected GATT Server.
in	*pChar*	Pointer to the Characteristic being written.
in	*valueLength*	Size in bytes of the value to be written.
in	*aValue*	Array of bytes to be written.
in	*aCsrk*	CSRK to be used for data signing.

Returns

gBleSuccess_c or error.

5.3 Typedef Documentation

5.3.1 typedef void(∗ gattClientProcedureCallback_t) (deviceId_t deviceId, gattProcedureType_t procedureType, gattProcedureResult_t procedureResult, bleResult_t error)

GATT Client Procedure Callback type.

5.3.2 typedef void(∗ gattClientNotificationCallback_t) (deviceId_t deviceId, uint16_t characteristicValueHandle, uint8_t ∗aValue, uint16_t valueLength)

GATT Client Notification Callback prototype.

5.3.3 typedef gattClientNotificationCallback_t gattClientIndicationCallback_t

GATT Client Indication Callback prototype.

5.4 Enumeration Type Documentation

5.4.1 enum gattProcedureType_t

GATT Client Procedure type.

Enumerator

gGattProcExchangeMtu_c	MTU Exchange.
gGattProcDiscoverAllPrimaryServices_c	Primary Service Discovery.
gGattProcDiscoverPrimaryServicesByUuid_c	Discovery of Services by UUID.
gGattProcFindIncludedServices_c	Discovery of Included Services within a Service range.
gGattProcDiscoverAllCharacteristics_c	Characteristic Discovery within Service range.
gGattProcDiscoverCharacteristicByUuid_c	Characteristic Discovery by UUID.
gGattProcDiscoverAllCharacteristicDescriptors_c	Characteristic Descriptor Discovery.
gGattProcReadCharacteristicValue_c	Characteristic Reading using Value handle.
gGattProcReadUsingCharacteristicUuid_c	Characteristic Reading by UUID.
gGattProcReadMultipleCharacteristicValues_c	Reading multiple Characteristics at once.
gGattProcWriteCharacteristicValue_c	Characteristic Writing.
gGattProcReadCharacteristicDescriptor_c	Reading Characteristic Descriptors.
gGattProcWriteCharacteristicDescriptor_c	Writing Characteristic Descriptors.

5.4.2 enum gattProcedureResult_t

GATT Client Procedure Result type.

Enumerator

gGattProcSuccess_c	The procedure was completed successfully.
gGattProcError_c	The procedure was terminated due to an error.

5.5 Function Documentation

5.5.1 bleResult_t GattClient_Init (void)

Initializes the GATT Client functionality.

Remarks

This should be called once at device startup, if necessary.
This function executes synchronously.

5.5.2 bleResult_t GattClient_ResetProcedure (void)

Resets any ongoing GATT Client Procedure.

Remarks

This function should be called if an ongoing Client procedure needs to be stopped.

5.5.3 bleResult_t GattClient_RegisterProcedureCallback (gattClientProcedure↩ Callback_t *callback*)

Installs the application callback for the GATT Client module Procedures.

Parameters

`in`	*callback*	Application defined callback to be triggered by this module.

Returns

gBleSuccess_c or error.

Remarks

This function executes synchronously.

5.5.4 bleResult_t GattClient_RegisterNotificationCallback (gattClientNotification↩ Callback_t *callback*)

Installs the application callback for Server Notifications.

Parameters

in	*callback*	Application defined callback to be triggered by this module.

Returns

gBleSuccess_c or error.

Remarks

This function executes synchronously.

5.5.5 bleResult_t GattClient_RegisterIndicationCallback (gattClientIndication↩ Callback_t *callback*)

Installs the application callback for Server Indications.

Parameters

in	*callback*	Application defined callback to be triggered by this module.

Returns

gBleSuccess_c or error.

Remarks

This function executes synchronously.

5.5.6 bleResult_t GattClient_ExchangeMtu (deviceId_t *deviceId,* uint16_t *mtu*)

Initializes the MTU Exchange procedure.

Parameters

in	*deviceId*	Device ID of the connected peer.
in	*mtu*	Desired MTU size.

Returns

gBleSuccess_c or error.

Remarks

If gBleSuccess_c is returned, the completion of this procedure is signalled by the Client Procedure callback.

5.5.7 bleResult_t GattClient_DiscoverAllPrimaryServices (deviceId_t *deviceId,* gattService_t ∗ *aOutPrimaryServices,* uint8_t *maxServiceCount,* uint8_t ∗ *pOutDiscoveredCount*)

Initializes the Primary Service Discovery procedure.

Parameters

in	*deviceId*	Device ID of the connected peer.
out	*aOutPrimary←Services*	Statically allocated array of gattService_t. The GATT module fills each Service's handle range and UUID.
in	*maxService←Count*	Maximum number of services to be filled.
out	*pOut←Discovered←Count*	The actual number of services discovered.

Returns

gBleSuccess_c or error.

Remarks

If gBleSuccess_c is returned, the completion of this procedure is signalled by the Client Procedure callback.

5.5.8 bleResult_t GattClient_DiscoverPrimaryServicesByUuid (deviceId_t *deviceId,* bleUuidType_t *uuidType,* const bleUuid_t ∗ *pUuid,* gattService_t ∗ *aOutPrimaryServices,* uint8_t *maxServiceCount,* uint8_t ∗ *pOutDiscoveredCount*)

Initializes the Primary Service Discovery By UUID procedure.

Parameters

in	*deviceId*	Device ID of the connected peer.
in	*uuidType*	Service UUID type.
in	*pUuid*	Service UUID.
out	*aOutPrimary←Services*	Statically allocated array of gattService_t. The GATT module fills each Service's handle range.
in	*maxService←Count*	Maximum number of services to be filled.
out	*pOut←Discovered←Count*	The actual number of services discovered.

Returns

gBleSuccess_c or error.

Remarks

If gBleSuccess_c is returned, the completion of this procedure is signalled by the Client Procedure callback.

5.5.9 bleResult_t GattClient_FindIncludedServices (deviceId_t *deviceId*, gattService_t ∗ *pIoService*, uint8_t *maxServiceCount*)

Initializes the Find Included Services procedure.

Parameters

`in`	*deviceId*	Device ID of the connected peer.
`in,out`	*pIoService*	The service within which inclusions should be searched. The GATT module uses the Service's handle range and fills the included Services' handle ranges, UUID types and the UUIDs if they are 16-bit UUIDs.
`in`	*maxService←Count*	Maximum number of included services to be filled.

Returns

gBleSuccess_c or error.

Remarks

If gBleSuccess_c is returned, the completion of this procedure is signalled by the Client Procedure callback.

5.5.10 bleResult_t GattClient_DiscoverAllCharacteristicsOfService (deviceId_t *deviceId*, gattService_t ∗ *pIoService*, uint8_t *maxCharacteristicCount*)

Initializes the Characteristic Discovery procedure for a given Service.

Parameters

`in`	*deviceId*	Device ID of the connected peer.
`in,out`	*pIoService*	The service within which characteristics should be searched. The GATT module uses the Characteristic's range.

in	*max←Characteristic←Count*	Maximum number of characteristics to be filled.

Returns

gBleSuccess_c or error.

Remarks

If gBleSuccess_c is returned, the completion of this procedure is signalled by the Client Procedure callback.

5.5.11 bleResult_t GattClient_DiscoverCharacteristicOfServiceByUuid (deviceId_t *deviceId,* bleUuidType_t *uuidType,* const bleUuid_t ∗ *pUuid,* const gattService_t ∗ *pService,* gattCharacteristic_t ∗ *aOutCharacteristics,* uint8_t *maxCharacteristicCount,* uint8_t ∗ *pOutDiscoveredCount*)

Initializes the Characteristic Discovery procedure for a given Service, with a given UUID.

Parameters

in	*deviceId*	Device ID of the connected peer.
in	*uuidType*	Characteristic UUID type.
in	*pUuid*	Characteristic UUID.
in	*pService*	The service within which characteristics should be searched.
out	*aOut←Characteristics*	The allocated array of Characteristics to be filled.
in	*max←Characteristic←Count*	Maximum number of characteristics to be filled.
out	*pOut←Discovered←Count*	The actual number of characteristics discovered.

Returns

gBleSuccess_c or error.

Remarks

If gBleSuccess_c is returned, the completion of this procedure is signalled by the Client Procedure callback.

5.5.12 bleResult_t GattClient_DiscoverAllCharacteristicDescriptors (deviceId_t *deviceId,* gattCharacteristic_t $*$ *ploCharacteristic,* uint16_t *endingHandle,* uint8_t *maxDescriptorCount*)

Initializes the Characteristic Descriptor Discovery procedure for a given Characteristic.

Parameters

in	*deviceId*	Device ID of the connected peer.
in,out	*pIo←Characteristic*	The characteristic within which descriptors should be searched. The G←ATT module uses the Characteristic's handle and fills each descriptor's handle and UUID.
in	*endingHandle*	The last handle of the Characteristic.
in	*max←Descriptor←Count*	Maximum number of descriptors to be filled.

Returns

gBleSuccess_c or error.

Remarks

If gBleSuccess_c is returned, the completion of this procedure is signalled by the Client Procedure callback. The endingHandle parameter should be known by the application if Characteristic Discovery was performed, i.e., if the next Characteristic declaration handle is known, then subtract 1 to obtain the endingHandle for the current Characteristic. If the last handle of the Characteristic is still unknown, set the endingHandle parameter to 0xFFFF.

5.5.13 bleResult_t GattClient_ReadCharacteristicValue (deviceId_t *deviceId,* gattCharacteristic_t ∗ *pIoCharacteristic,* uint16_t *maxReadBytes*)

Initializes the Characteristic Read procedure for a given Characteristic.

Parameters

in	*deviceId*	Device ID of the connected peer.
in,out	*pIo←Characteristic*	The characteristic whose value must be read. The GATT module uses the value handle and fills the value and length.
in	*maxReadBytes*	Maximum number of bytes to be read.

Returns

gBleSuccess_c or error.

Remarks

If gBleSuccess_c is returned, the completion of this procedure is signalled by the Client Procedure callback.

5.5.14 bleResult_t GattClient_ReadUsingCharacteristicUuid (deviceId_t *deviceId*, bleUuidType_t *uuidType*, const bleUuid_t ∗ *pUuid*, const gattHandleRange_t ∗ *pHandleRange*, uint8_t ∗ *aOutBuffer*, uint16_t *maxReadBytes*, uint16_t ∗ *pOutActualReadBytes*)

Initializes the Characteristic Read By UUID procedure.

Parameters

in	*deviceId*	Device ID of the connected peer.
in	*uuidType*	Characteristic UUID type.
in	*pUuid*	Characteristic UUID.
in	*pHandleRange*	Handle range for the search or NULL. If this is NULL, the search range is 0x0001-0xffff.
out	*aOutBuffer*	The allocated buffer to read into.
in	*maxReadBytes*	Maximum number of bytes to be read.
out	*pOutActual←ReadBytes*	The actual number of bytes read.

Returns

gBleSuccess_c or error.

Remarks

This procedure returns the Characteristics found within the specified range with the specified UU←ID. aOutBuffer will contain the Handle-Value pair length (1 byte), then Handle-Value pairs for all Characteristic Values found with the specified UUID.
If gBleSuccess_c is returned, the completion of this procedure is signalled by the Client Procedure callback.

5.5.15 bleResult_t GattClient_ReadMultipleCharacteristicValues (deviceId_t *deviceId,* uint8_t *cNumCharacteristics,* gattCharacteristic_t ∗ *aIoCharacteristics*)

Initializes the Characteristic Read Multiple procedure.

Parameters

in	*deviceId*	Device ID of the connected peer.
in,out	*aIo←Characteristics*	Array of the characteristics whose values are to be read. The GA←TT module uses each Characteristic's value handle and maxValueLength fills each value and length.
in	*cNum←Characteristics*	Number of characteristics in the array.

Returns

gBleSuccess_c or error.

Remarks

If gBleSuccess_c is returned, the completion of this procedure is signalled by the Client Procedure callback.

5.5.16 bleResult_t GattClient_WriteCharacteristicValue (deviceId_t *deviceId,* const gattCharacteristic_t ∗ *pCharacteristic,* uint16_t *valueLength,* const uint8_t ∗ *aValue,* bool_t *withoutResponse,* bool_t *signedWrite,* bool_t *doReliableLongCharWrites,* const uint8_t ∗ *aCsrk*)

Initializes the Characteristic Write procedure for a given Characteristic.

Parameters

in	*deviceId*	Device ID of the connected peer.
in	*pCharacteristic*	The characteristic whose value must be written. The GATT module uses the value handle.
in	*valueLength*	Number of bytes to be written.
in	*aValue*	Array of bytes to be written.
in	*without↩ Response*	Indicates if a Write Command is used.
in	*signedWrite*	Indicates if a Signed Write is performed.
in	*doReliable↩ LongChar↩ Writes*	Indicates Reliable Long Writes.
in	*aCsrk*	The CSRK (gcCsrkSize_d bytes) if signedWrite is TRUE, ignored otherwise.

Returns

gBleSuccess_c or error.

Remarks

If gBleSuccess_c is returned, the completion of this procedure is signalled by the Client Procedure callback.

5.5.17 bleResult_t GattClient_ReadCharacteristicDescriptor (deviceId_t *deviceId,* gattAttribute_t ∗ *pIoDescriptor,* uint16_t *maxReadBytes*)

Initializes the Characteristic Descriptor Read procedure for a given Characteristic Descriptor.

Parameters

`in`	*deviceId*	Device ID of the connected peer.
`in,out`	*pIoDescriptor*	The characteristic descriptor whose value must be read. The GA↵ TT module uses the attribute's handle and fills the attribute's value and length.
`in`	*maxReadBytes*	Maximum number of bytes to be read.

Returns

gBleSuccess_c or error.

Remarks

If gBleSuccess_c is returned, the completion of this procedure is signalled by the Client Procedure callback.

5.5.18 bleResult_t GattClient_WriteCharacteristicDescriptor (deviceId_t *deviceId*, const gattAttribute_t ∗ *pDescriptor*, uint16_t *valueLength*, const uint8_t ∗ *aValue*)

Initializes the Characteristic Descriptor Write procedure for a given Characteristic Descriptor.

Parameters

`in`	*deviceId*	Device ID of the connected peer.
`in`	*pDescriptor*	The characteristic descriptor whose value must be written. The GATT module uses the attribute's handle.
`in`	*valueLength*	Number of bytes to be written.
`in`	*aValue*	Array of bytes to be written.

Returns

gBleSuccess_c or error.

Remarks

If gBleSuccess_c is returned, the completion of this procedure is signalled by the Client Procedure callback.

Chapter 6
GATT - Server APIs

6.1 Overview

Files

- file gatt_server_interface.h

Data Structures

- struct gattServerMtuChangedEvent_t
- struct gattServerAttributeWrittenEvent_t
- struct gattServerLongCharacteristicWrittenEvent_t
- struct gattServerCccdWrittenEvent_t
- struct gattServerAttributeReadEvent_t
- struct gattServerProcedureError_t
- struct gattServerInvalidPdu_t
- struct gattServerEvent_t
- union gattServerEvent_t.eventData

Typedefs

- typedef void(∗ gattServerCallback_t) (deviceId_t deviceId, gattServerEvent_t ∗pServerEvent)

Enumerations

- enum gattServerEventType_t {
  gEvtMtuChanged_c,
  gEvtHandleValueConfirmation_c,
  gEvtAttributeWritten_c,
  gEvtCharacteristicCccdWritten_c,
  gEvtAttributeWrittenWithoutResponse_c,
  gEvtError_c,
  gEvtLongCharacteristicWritten_c,
  gEvtAttributeRead_c,
  gEvtInvalidPduReceived_c }
- enum gattServerProcedureType_t {
  gSendAttributeWrittenStatus_c,
  gSendAttributeReadStatus_c,
  gSendNotification_c,
  gSendIndication_c }

Functions

- bleResult_t GattServer_Init (void)
- bleResult_t GattServer_RegisterCallback (gattServerCallback_t callback)
- bleResult_t GattServer_RegisterHandlesForWriteNotifications (uint8_t handleCount, const uint16↩_t ∗aAttributeHandles)
- bleResult_t GattServer_UnregisterHandlesForWriteNotifications (uint8_t handleCount, const uint16_t ∗aAttributeHandles)
- bleResult_t GattServer_SendAttributeWrittenStatus (deviceId_t deviceId, uint16_t attributeHandle, uint8_t status)
- bleResult_t GattServer_RegisterHandlesForReadNotifications (uint8_t handleCount, const uint16↩_t ∗aAttributeHandles)
- bleResult_t GattServer_UnregisterHandlesForReadNotifications (uint8_t handleCount, const uint16_t ∗aAttributeHandles)
- bleResult_t GattServer_SendAttributeReadStatus (deviceId_t deviceId, uint16_t attributeHandle, uint8_t status)
- bleResult_t GattServer_SendNotification (deviceId_t deviceId, uint16_t handle)
- bleResult_t GattServer_SendIndication (deviceId_t deviceId, uint16_t handle)
- bleResult_t GattServer_SendInstantValueNotification (deviceId_t deviceId, uint16_t handle, uint16_t valueLength, const uint8_t ∗aValue)
- bleResult_t GattServer_SendInstantValueIndication (deviceId_t deviceId, uint16_t handle, uint16↩_t valueLength, const uint8_t ∗aValue)
- bleResult_t GattServer_RegisterUniqueHandlesForNotifications (bool_t bWrite, bool_t bRead)

6.2 Data Structure Documentation

6.2.1 struct gattServerMtuChangedEvent_t

GATT Server MTU Changed Event structure.

Data Fields

uint16_t	newMtu	Value of the agreed ATT_MTU for this connection.

6.2.2 struct gattServerAttributeWrittenEvent_t

GATT Server Attribute Written Event structure.

Data Fields

uint16_t	handle	Handle of the attribute.
uint16_t	cValueLength	Length of the attribute value array.
uint8_t ∗	aValue	Attribute value array attempted to be written.

6.2.3 struct gattServerLongCharacteristicWrittenEvent_t

GATT Server Long Characteristic Written Event structure.

Data Fields

uint16_t	handle	Handle of the Characteristic Value.
uint16_t	cValueLength	Length of the value written.
uint8_t *	aValue	Pointer to the attribute value in the database.

6.2.4 struct gattServerCccdWrittenEvent_t

GATT Server CCCD Written Event structure.

Data Fields

uint16_t	handle	Handle of the CCCD attribute.
gattCccd←Flags_t	newCccd	New value of the CCCD.

6.2.5 struct gattServerAttributeReadEvent_t

GATT Server Attribute Read Event structure.

Data Fields

uint16_t	handle	Handle of the attribute.

6.2.6 struct gattServerProcedureError_t

Server-initiated procedure error structure.

Data Fields

gattServer←Procedure←Type_t	procedureType	Procedure that generated error.
bleResult_t	error	Error generated.

6.2.7 struct gattServerInvalidPdu_t

ATT PDU that generated the error.

Data Fields

attOpcode_t	attOpCode	The invalid ATT op code that generated the error.

6.2.8 struct gattServerEvent_t

GATT Server Event structure: type + data.

Data Fields

gattServer↩EventType_t	eventType	Event type.
union gatt↩ServerEvent_t	eventData	Event data : selected according to event type.

6.2.9 union gattServerEvent_t.eventData

Data Fields

gattServer↩MtuChanged↩Event_t	mtuChanged↩Event	For event type gEvtMtuChanged_c: the new value of the ATT_↩MTU.
gattServer↩Attribute↩WrittenEvent↩_t	attribute↩WrittenEvent	For event types gEvtAttributeWritten_c, gEvtAttributeWritten↩WithoutResponse_c: handle and value of the attempted write.
gattServer↩CccdWritten↩Event_t	charCccd↩WrittenEvent	For event type gEvtCharacteristicCccdWritten_c: handle and value of the CCCD.
gattServer↩Procedure↩Error_t	procedureError	For event type gEvtError_c: error that terminated a Server-initiated procedure.
gattServer↩Long↩Characteristic↩WrittenEvent↩_t	longChar↩WrittenEvent	For event type gEvtLongCharacteristicWritten_c: handle and value.

gattServer←Attribute←ReadEvent_t	attributeRead←Event	For event types gEvtAttributeRead_c: handle of the attempted read.
gattServer←InvalidPdu_t	attributeOp←Code	For event type gEvtInvalidPduReceived_c: the ATT PDU that generated the error.

6.3 Typedef Documentation

6.3.1 typedef void(∗ gattServerCallback_t) (deviceId_t deviceId, gattServerEvent_t ∗pServerEvent)

GATT Server Callback prototype.

6.4 Enumeration Type Documentation

6.4.1 enum gattServerEventType_t

GATT Server Event type enumeration.

Enumerator

gEvtMtuChanged_c ATT_MTU was changed after the MTU exchange.

gEvtHandleValueConfirmation_c Received a Handle Value Confirmation from the Client.

gEvtAttributeWritten_c An attribute registered with GattServer_RegisterHandlesForWrite←Notifications was written. After receiving this event, application must call GattServer_←SendAttributeWrittenStatus. Application must write the Attribute in the Database if it considers necessary.

gEvtCharacteristicCccdWritten_c A CCCD was written. Application should save the CCCD value with Gap_SaveCccd.

gEvtAttributeWrittenWithoutResponse_c An attribute registered with GattServer_Register←HandlesForWriteNotifications was written without response (with ATT Write Command). Application must write the Attribute Value in the Database if it considers necessary.

gEvtError_c An error appeared during a Server-initiated procedure.

gEvtLongCharacteristicWritten_c A long characteristic was written.

gEvtAttributeRead_c An attribute registered with GattServer_RegisterHandlesForReadNotifications is being read. After receiving this event, application must call GattServer_SendAttributeRead←Status.

gEvtInvalidPduReceived_c An invalid PDU was received from Client. Application decides if disconnection is required

6.4.2 enum gattServerProcedureType_t

Server-initiated procedure type enumeration.

Enumerator

gSendAttributeWrittenStatus_c	Procedure initiated by GattServer_SendAttributeWrittenStatus.
gSendAttributeReadStatus_c	Procedure initiated by GattServer_SendAttributeReadStatus.
gSendNotification_c	Procedure initiated by GattServer_SendNotification.
gSendIndication_c	Procedure initiated by GattServer_SendIndication.

6.5 Function Documentation

6.5.1 bleResult_t GattServer_Init (void)

Initializes the GATT Server module.

Returns

gBleSuccess_c or error.

Remarks

Application does not need to call this function if Gatt_Init() is called.
This function executes synchronously.

6.5.2 bleResult_t GattServer_RegisterCallback (gattServerCallback_t *callback*)

Installs an application callback for the GATT Server module.

Parameters

in	*callback*	Application-defined callback to be triggered by this module.

Returns

gBleSuccess_c or error.

Remarks

This function executes synchronously.

6.5.3 bleResult_t GattServer_RegisterHandlesForWriteNotifications (uint8_t *handleCount*, const uint16_t ∗ *aAttributeHandles*)

Registers the attribute handles that will be notified through the GATT Server callback when a GATT Client attempts to modify the attributes' values.

Parameters

in	*handleCount*	Number of handles in array.
in	*aAttribute←Handles*	Array of handles.

Returns

gBleSuccess_c or error.

Remarks

The application is responsible for actually writing the new requested values in the GATT database. Service and profile-specific control-point characteristics should have their value handles in this list so that the application may get notified when a GATT Client writes it.
This function executes synchronously.

6.5.4 bleResult_t GattServer_UnregisterHandlesForWriteNotifications (uint8_t *handleCount*, const uint16_t ∗ *aAttributeHandles*)

Unregisters the attribute handles that will be notified through the GATT Server callback when a GATT Client attempts to modify the attributes' values.

Parameters

in	*handleCount*	Number of handles in array.
in	*aAttribute←Handles*	Array of handles.

Returns

gBleSuccess_c or error.

Remarks

To unregister all the list, pass 0 count and NULL.
This function executes synchronously.

6.5.5 bleResult_t GattServer_SendAttributeWrittenStatus (deviceId_t *deviceId*, uint16_t *attributeHandle*, uint8_t *status*)

Responds to an intercepted attribute write operation.

Parameters

in	*deviceId*	The device ID of the connected peer.
in	*attribute↩ Handle*	The attribute handle that was written.
in	*status*	The status of the write operation. If this parameter is equal to gAttErr↩ CodeNoError_c then an ATT Write Response will be sent to the peer. Else an ATT Error Response with the provided status will be sent to the peer.

Remarks

This function must be called by the application when receiving the gEvtAttributeWritten_c Server event. The status value may contain application- or profile-defined error codes.

6.5.6 bleResult_t GattServer_RegisterHandlesForReadNotifications (uint8_t *handleCount,* const uint16_t ∗ *aAttributeHandles*)

Registers the attribute handles that will be notified through the GATT Server callback when a GATT Client attempts to read the attributes' values.

Parameters

in	*handleCount*	Number of handles in array.
in	*aAttribute↩ Handles*	Array of handles.

Returns

gBleSuccess_c or error.

Remarks

The application may modify the attribute's value in the GATT Database before sending the response with GattServer_SendAttributeReadStatus.
This function executes synchronously.

6.5.7 bleResult_t GattServer_UnregisterHandlesForReadNotifications (uint8_t *handleCount,* const uint16_t ∗ *aAttributeHandles*)

Unregisters the attribute handles that will be notified through the GATT Server callback when a GATT Client attempts to read the attributes' values.

Parameters

in	*handleCount*	Number of handles in array.
in	*aAttribute←Handles*	Array of handles.

Returns

gBleSuccess_c or error.

Remarks

To unregister all the list, pass 0 count and NULL.
This function executes synchronously.

6.5.8 bleResult_t GattServer_SendAttributeReadStatus (deviceId_t *deviceId,* uint16_t *attributeHandle,* uint8_t *status*)

Responds to an intercepted attribute read operation.

Parameters

in	*deviceId*	The device ID of the connected peer.
in	*attribute←Handle*	The attribute handle that was being read.
in	*status*	The status of the read operation. If this parameter is equal to gAttErr←CodeNoError_c then an ATT Read Response will be sent to the peer containing the attribute value from the GATT Database. Else an ATT Error Response with the provided status will be sent to the peer.

Remarks

This function must be called by the application when receiving the gEvtAttributeRead_c Server event. The status value may contain application- or profile-defined error codes.

6.5.9 bleResult_t GattServer_SendNotification (deviceId_t *deviceId,* uint16_t *handle*)

Sends a notification to a peer GATT Client using the Characteristic Value from the GATT Database.

Parameters

`in`	*deviceId*	The device ID of the connected peer.
`in`	*handle*	Handle of the Value of the Characteristic to be notified.

Returns

gBleSuccess_c or error.

6.5.10 bleResult_t GattServer_SendIndication (deviceId_t *deviceId,* uint16_t *handle*)

Sends an indication to a peer GATT Client using the Characteristic Value from the GATT Database.

Parameters

`in`	*deviceId*	The device ID of the connected peer.
`in`	*handle*	Handle of the Value of the Characteristic to be indicated.

Returns

gBleSuccess_c or error.

6.5.11 bleResult_t GattServer_SendInstantValueNotification (deviceId_t *deviceId,* uint16_t *handle,* uint16_t *valueLength,* const uint8_t ∗ *aValue*)

Sends a notification to a peer GATT Client with data given as parameter, ignoring the GATT Database.

Parameters

`in`	*deviceId*	The device ID of the connected peer.
`in`	*handle*	Handle of the Value of the Characteristic to be notified.
`in`	*valueLength*	Length of data to be notified.
`in`	*aValue*	Data to be notified.

Returns

gBleSuccess_c or error.

6.5.12 bleResult_t GattServer_SendInstantValueIndication (deviceId_t *deviceId,* uint16_t *handle,* uint16_t *valueLength,* const uint8_t ∗ *aValue*)

Sends an indication to a peer GATT Client with data given as parameter, ignoring the GATT Database.

Parameters

in	*deviceId*	The device ID of the connected peer.
in	*handle*	Handle of the Value of the Characteristic to be indicated.
in	*valueLength*	Length of data to be indicated.
in	*aValue*	Data to be indicated.

Returns

gBleSuccess_c or error.

6.5.13 bleResult_t GattServer_RegisterUniqueHandlesForNotifications (bool_t *bWrite*, bool_t *bRead*)

Registers all GATT DB dynamic attribute handles with unique value buffers to be notified through the GATT Server callback when a GATT Client attempts to read/write the attributes' values.

Parameters

in	*bWrite*	Enables/Disables write notifications.
in	*bRead*	Enables/Disables read notifications.

Returns

gBleSuccess_c or error.

Remarks

This function executes synchronously.
This function should be called when adding GATT DB unique value buffer characteristics or descriptors.

Chapter 7
GATT_DB - GATT Database Interface and Definitions

7.1 Overview

Files

- file gatt_database.h
- file gatt_db_app_interface.h

Data Structures

- struct gattDbAttribute_t

Macros

- #define gGattDbInvalidHandleIndex_d
- #define gGattDbInvalidHandle_d
- #define gPermissionNone_c
- #define gPermissionFlagReadable_c
- #define gPermissionFlagReadWithEncryption_c
- #define gPermissionFlagReadWithAuthentication_c
- #define gPermissionFlagReadWithAuthorization_c
- #define gPermissionFlagWritable_c
- #define gPermissionFlagWriteWithEncryption_c
- #define gPermissionFlagWriteWithAuthentication_c
- #define gPermissionFlagWriteWithAuthorization_c

Typedefs

- typedef uint8_t gattCharacteristicPropertiesBitFields_t
- typedef uint8_t gattAttributePermissionsBitFields_t

Enumerations

- enum gattCharacteristicPropertiesBitFields_tag {
  gGattCharPropNone_c,
  gGattCharPropBroadcast_c,
  gGattCharPropRead_c,
  gGattCharPropWriteWithoutRsp_c,
  gGattCharPropWrite_c,
  gGattCharPropNotify_c,
  gGattCharPropIndicate_c,
  gGattCharPropAuthSignedWrites_c,
  gGattCharPropExtendedProperties_c }

- enum gattDbAccessType_t {
gAccessRead_c,
gAccessWrite_c,
gAccessNotify_c }

Functions

- uint16_t GattDb_GetIndexOfHandle (uint16_t handle)
- bleResult_t GattDb_Init (void)
- bleResult_t GattDb_WriteAttribute (uint16_t handle, uint16_t valueLength, const uint8_t ∗aValue)
- bleResult_t GattDb_ReadAttribute (uint16_t handle, uint16_t maxBytes, uint8_t ∗aOutValue, uint16_t ∗pOutValueLength)
- bleResult_t GattDb_FindServiceHandle (uint16_t startHandle, bleUuidType_t serviceUuidType, const bleUuid_t ∗pServiceUuid, uint16_t ∗pOutServiceHandle)
- bleResult_t GattDb_FindCharValueHandleInService (uint16_t serviceHandle, bleUuidType_↩ t characteristicUuidType, const bleUuid_t ∗pCharacteristicUuid, uint16_t ∗pOutCharValueHandle)
- bleResult_t GattDb_FindCccdHandleForCharValueHandle (uint16_t charValueHandle, uint16_↩ t ∗pOutCccdHandle)
- bleResult_t GattDb_FindDescriptorHandleForCharValueHandle (uint16_t charValueHandle, ble↩ UuidType_t descriptorUuidType, const bleUuid_t ∗pDescriptorUuid, uint16_t ∗pOutDescriptor↩ Handle)

Variables

- uint16_t gGattDbAttributeCount_c
- gattDbAttribute_t ∗ gattDatabase

7.2 Data Structure Documentation

7.2.1 struct gattDbAttribute_t

Attribute structure.

Data Fields

uint16_t	handle	The attribute handle - cannot be 0x0000. The attribute handles need not be consecutive, but must be strictly increasing.
uint16_t	permissions	Attribute permissions as defined by the ATT.
uint32_t	uuid	The UUID should be read according to the gattDbAttribute_t.↩ uuidType member: for 2-byte and 4-byte UUIDs, this contains the value of the UUID; for 16-byte UUIDs, this is a pointer to the allocated 16-byte array containing the UUID.

uint8_t *	pValue	A pointer to allocated value array.
uint16_t	valueLength	The size of the value array.
uint16_t	uuidType: 2	Identifies the length of the UUID; values interpreted according to the bleUuidType_t enumeration.
uint16_t	maxVariable↩ ValueLength: 10	The maximum length of the attribute value array; if this is set to 0, then the attribute's length is fixed and cannot be changed.

7.3 Macro Definition Documentation

7.3.1 #define gGattDbInvalidHandleIndex_d

Special value returned by GattDb_GetIndexOfHandle to signal that an invalid attribute handle was given as parameter.

7.3.2 #define gGattDbInvalidHandle_d

Special value used to mark an invalid attribute handle.

Attribute handles are strictly positive.

7.3.3 #define gPermissionNone_c

No permissions selected.

7.3.4 #define gPermissionFlagReadable_c

Attribute can be read.

7.3.5 #define gPermissionFlagReadWithEncryption_c

Attribute may be read only if link is encrypted.

7.3.6 #define gPermissionFlagReadWithAuthentication_c

Attribute may be read only by authenticated peers.

7.3.7 #define gPermissionFlagReadWithAuthorization_c

Attribute may be read only by authorized peers.

7.3.8 #define gPermissionFlagWritable_c

Attribute can be written.

7.3.9 #define gPermissionFlagWriteWithEncryption_c

Attribute may be written only if link is encrypted.

7.3.10 #define gPermissionFlagWriteWithAuthentication_c

Attribute may be written only by authenticated peers.

7.3.11 #define gPermissionFlagWriteWithAuthorization_c

Attribute may be written only by authorized peers.

7.4 Typedef Documentation

7.4.1 typedef uint8_t gattCharacteristicPropertiesBitFields_t

Bit fields for Characteristic properties.

7.4.2 typedef uint8_t gattAttributePermissionsBitFields_t

Bit fields for attribute permissions.

7.5 Enumeration Type Documentation

7.5.1 enum gattCharacteristicPropertiesBitFields_tag

Enumerator

gGattCharPropNone_c	No Properties selected.
gGattCharPropBroadcast_c	Characteristic can be broadcast.
gGattCharPropRead_c	Characteristic can be read.
gGattCharPropWriteWithoutRsp_c	Characteristic can be written without response.

gGattCharPropWrite_c	Characteristic can be written with response.
gGattCharPropNotify_c	Characteristic can be notified.
gGattCharPropIndicate_c	Characteristic can be indicated.
gGattCharPropAuthSignedWrites_c	Characteristic can be written with signed data.
gGattCharPropExtendedProperties_c	Extended Characteristic properties.

7.5.2 enum gattDbAccessType_t

Attribute access type.

7.6 Function Documentation

7.6.1 uint16_t GattDb_GetIndexOfHandle (uint16_t *handle*)

Returns the database index for a given attribute handle.

Parameters

`in`	*handle*	The attribute handle.

Returns

The index of the given attribute in the database or gGattDbInvalidHandleIndex_d.

7.6.2 bleResult_t GattDb_Init (void)

Initializes the GATT database at runtime.

Remarks

This function should be called only once at device start-up. In the current stack implementation, it is called internally by Ble_HostInitialize.
This function executes synchronously.

Returns

gBleSuccess_c or error.

7.6.3 bleResult_t GattDb_WriteAttribute (uint16_t *handle,* uint16_t *valueLength,* const uint8_t ∗ *aValue*)

Writes an attribute from the application level.

This function can be called by the application code to modify an attribute in the database. It should only be used by the application to modify a Characteristic's value based on the application logic (e.g., external sensor readings).

Parameters

`in`	*handle*	The handle of the attribute to be written.
`in`	*valueLength*	The number of bytes to be written.
`in`	*aValue*	The source buffer containing the value to be written.

Returns

gBleSuccess_c or error.

Remarks

This function executes synchronously.

7.6.4 bleResult_t GattDb_ReadAttribute (uint16_t *handle,* uint16_t *maxBytes,* uint8_t ∗ *aOutValue,* uint16_t ∗ *pOutValueLength*)

Reads an attribute from the application level.

This function can be called by the application code to read an attribute in the database.

Parameters

`in`	*handle*	The handle of the attribute to be read.
`in`	*maxBytes*	The maximum number of bytes to be received.
`out`	*aOutValue*	The pre-allocated buffer ready to receive the bytes.
`out`	*pOutValue←Length*	The actual number of bytes received.

Returns

gBleSuccess_c or error.

Remarks

This function executes synchronously.

7.6.5 bleResult_t GattDb_FindServiceHandle (uint16_t *startHandle,* bleUuidType_t *serviceUuidType,* const bleUuid_t ∗ *pServiceUuid,* uint16_t ∗ *pOutServiceHandle*)

Finds the handle of a Service Declaration with a given UUID inside the database.

Parameters

in	*startHandle*	The handle to start the search. Should be 0x0001 on the first call.
in	*serviceUuid←Type*	Service UUID type.
in	*pServiceUuid*	Service UUID.
out	*pOutService←Handle*	Pointer to the service declaration handle to be written.

Returns

gBleSuccess_c or error.

Return values

gBleSuccess_c	Service Declaration found, handle written in pOutCharValueHandle.
gGattDbInvalidHandle_c	Invalid Start Handle.
gGattDbServiceNot←Found_c	Service with given UUID not found.

Remarks

This function executes synchronously.
The startHandle should be set to 0x0001 when this function is called for the first time. If multiple Services with the same UUID are expected, then after the first successful call the function may be called again with the startHandle equal to the found service handle plus one.

7.6.6 bleResult_t GattDb_FindCharValueHandleInService (uint16_t *serviceHandle,* bleUuidType_t *characteristicUuidType,* const bleUuid_t ∗ *pCharacteristicUuid,* uint16_t ∗ *pOutCharValueHandle*)

Finds the handle of a Characteristic Value with a given UUID inside a Service.

The Service is input by its declaration handle.

Parameters

in	*serviceHandle*	The handle of the Service declaration.
in	*characteristic←UuidType*	Characteristic UUID type.

in	*p↩ Characteristic↩ Uuid*	Characteristic UUID.
out	*pOutChar↩ ValueHandle*	Pointer to the characteristic value handle to be written.

Returns

gBleSuccess_c or error.

Return values

gBleSuccess_c	Characteristic Value found, handle written in pOutCharValueHandle.
gGattDbInvalidHandle_c	Handle not found or not a Service declaration.
gGattDbCharacteristic↩ NotFound_c	Characteristic Value with given UUID not found.

Remarks

This function executes synchronously.

7.6.7 bleResult_t GattDb_FindCccdHandleForCharValueHandle (uint16_t *charValueHandle,* uint16_t ∗ *pOutCccdHandle*)

Finds the handle of a Characteristic's CCCD given the Characteristic's Value handle.

Parameters

in	*charValue↩ Handle*	The handle of the Service declaration.
out	*pOutCccd↩ Handle*	Pointer to the CCCD handle to be written.

Returns

gBleSuccess_c or error.

Return values

gBleSuccess_c	CCCD found, handle written in pOutCccdHandle.

gGattDbInvalidHandle_c	Invalid Characteristic Value handle.
gGattDbCccdNotFound←_c	CCCD not found for this Characteristic.

Remarks

This function executes synchronously.

7.6.8 bleResult_t GattDb_FindDescriptorHandleForCharValueHandle (uint16_t *charValueHandle,* bleUuidType_t *descriptorUuidType,* const bleUuid_t ∗ *pDescriptorUuid,* uint16_t ∗ *pOutDescriptorHandle*)

Finds the handle of a Characteristic Descriptor given the Characteristic's Value handle and Descriptor's UUID.

Parameters

`in`	*charValue←Handle*	The handle of the Service declaration.
`in`	*descriptor←UuidType*	Descriptor's UUID type.
`in`	*pDescriptor←Uuid*	Descriptor's UUID.
`out`	*pOut←Descriptor←Handle*	Pointer to the Descriptor handle to be written.

Returns

gBleSuccess_c or error.

Return values

gBleSuccess_c	Descriptor found, handle written in pOutDescriptorHandle.
gGattDbInvalidHandle_c	Invalid Characteristic Value handle.
gGattDbDescriptorNot←Found_c	Descriptor not found for this Characteristic.

Remarks

This function executes synchronously.

7.7 Variable Documentation

7.7.1 uint16_t gGattDbAttributeCount_c

The number of attributes in the GATT Database.

7.7.2 gattDbAttribute_t∗ gattDatabase

Reference to the GATT database.

Chapter 8
L2CA

8.1 Overview

Files

- file l2ca_cb_interface.h
- file l2ca_types.h

Data Structures

- struct l2caLeCbConnectionRequest_t
- struct l2caLeCbConnectionComplete_t
- struct l2caLeCbDisconnection_t
- struct l2caLeCbNoPeerCredits_t
- struct l2caLeCbLocalCreditsNotification_t
- struct l2caLeCbError_t
- struct l2capControlMessage_t
- union l2capControlMessage_t.messageData

Macros

- #define **gL2capCidNull_c**
- #define **gL2capCidAtt_c**
- #define **gL2capCidSignaling_c**
- #define **gL2capCidSmp_c**
- #define **gL2capCidSigAssignedFirst_c**
- #define **gL2capCidSigAssignedLast_c**
- #define **gL2capCidLePsmDynamicFirst_c**
- #define **gL2capCidLePsmDynamicLast_c**
- #define **gL2capCidNotApplicable_c**
- #define **gL2caLePsmSigAssignedFirst_c**
- #define **gL2caLePsmSigAssignedLast_c**
- #define **gL2caLePsmDynamicFirst_c**
- #define **gL2caLePsmDynamicLast_c**
- #define **gL2capDefaultMtu_c**
- #define **gL2capDefaultMps_c**
- #define **gL2capMaximumMps_c**
- #define **gL2capHeaderLength_c**
- #define **gExpandAsEnum_m**(a, b, c)
- #define **gExpandAsTable_m**(a, b, c)
- #define **gLePsmSigAssignedNumbersTable_m**(entry)

Typedefs

- typedef void(∗ **l2caLeCbDataCallback_t**) (deviceId_t deviceId, uint16_t channelId, uint8_t ∗p↩ Packet, uint16_t packetLength)

- typedef void(∗ **l2caControlCallback_t**) (l2capControlMessage_t ∗pMessage)
- typedef l2caControlCallback_t **l2caLeCbControlCallback_t**
- typedef void(∗ **l2caGenericCallback_t**) (deviceId_t deviceId, uint8_t ∗pPacket, uint16_t packet↩ Length)

Enumerations

- enum **l2caLeCbConnectionRequestResult_t** {
  gSuccessful_c,
  gLePsmNotSupported_c,
  gNoResourcesAvailable_c,
  gInsufficientAuthentication_c,
  gInsufficientAuthorization_c,
  gInsufficientEncryptionKeySize_c,
  gInsufficientEncryption_c,
  gInvalidSourceCid_c,
  gSourceCidAreadyAllocated_c,
  gInvalidParameters_c,
  gCommandRejected_c,
  gResponseTimeout_c }
- enum **l2caErrorSource_t** {
  gL2ca_CancelConnection_c,
  gL2ca_SendLeFlowControlCredit_c,
  gL2ca_DisconnectLePsm_c,
  gL2ca_HandleSendLeCbData_c,
  gL2ca_HandleRecvLeCbData_c,
  gL2ca_HandleLeFlowControlCredit_c }
- enum **l2capControlMessageType_t** {
  gL2ca_LePsmConnectRequest_c,
  gL2ca_LePsmConnectionComplete_c,
  gL2ca_LePsmDisconnectNotification_c,
  gL2ca_NoPeerCredits_c,
  gL2ca_LocalCreditsNotification_c,
  gL2ca_Error_c }

Functions

- bleResult_t L2ca_RegisterLeCbCallbacks (l2caLeCbDataCallback_t pCallback, l2caLeCb↩ ControlCallback_t pCtrlCallback)
- bleResult_t L2ca_RegisterLePsm (uint16_t lePsm, uint16_t lePsmMtu)
- bleResult_t L2ca_DeregisterLePsm (uint16_t lePsm)
- bleResult_t L2ca_ConnectLePsm (uint16_t lePsm, deviceId_t deviceId, uint16_t initialCredits)
- bleResult_t L2ca_DisconnectLeCbChannel (deviceId_t deviceId, uint16_t channelId)
- bleResult_t L2ca_CancelConnection (uint16_t lePsm, deviceId_t deviceId, l2caLeCbConnection↩ RequestResult_t refuseReason)
- bleResult_t L2ca_SendLeCbData (deviceId_t deviceId, uint16_t channelId, const uint8_t ∗pPacket, uint16_t packetLength)
- bleResult_t L2ca_SendLeCredit (deviceId_t deviceId, uint16_t channelId, uint16_t credits)

8.2 Data Structure Documentation

8.2.1 struct l2caLeCbConnectionRequest_t

Data Fields

deviceId_t	deviceId	
uint16_t	lePsm	
uint16_t	peerMtu	
uint16_t	peerMps	
uint16_t	initialCredits	

8.2.2 struct l2caLeCbConnectionComplete_t

Data Fields

deviceId_t	deviceId	
uint16_t	cId	
uint16_t	peerMtu	
uint16_t	peerMps	
uint16_t	initialCredits	
l2caLeCb↩ Connection↩ Request↩ Result_t	result	

8.2.3 struct l2caLeCbDisconnection_t

Data Fields

deviceId_t	deviceId	
uint16_t	cId	

8.2.4 struct l2caLeCbNoPeerCredits_t

Data Fields

deviceId_t	deviceId	
uint16_t	cId	

8.2.5 struct l2caLeCbLocalCreditsNotification_t

Data Fields

deviceId_t	deviceId	
uint16_t	cId	
uint16_t	localCredits	

8.2.6 struct l2caLeCbError_t

Data Fields

deviceId_t	deviceId	
bleResult_t	result	
l2caError↩ Source_t	errorSource	

8.2.7 struct l2capControlMessage_t

Data Fields

l2capControl↩ MessageType↩ _t	messageType	
union l2capControl↩ Message_t	messageData	

8.2.8 union l2capControlMessage_t.messageData

Data Fields

l2caLeCb↩ Connection↩ Request_t	connection↩ Request	
l2caLeCb↩ Connection↩ Complete_t	connection↩ Complete	

l2caLeCb↩ Disconnection↩ _t	disconnection	
l2caLeCbNo↩ PeerCredits_t	noPeerCredits	
l2caLeCb↩ LocalCredits↩ Notification_t	localCredits↩ Notification	
l2caLeCb↩ Error_t	error	

8.3 Function Documentation

8.3.1 bleResult_t L2ca_RegisterLeCbCallbacks (l2caLeCbDataCallback_t *pCallback,* l2caLeCbControlCallback_t *pCtrlCallback*)

Registers callbacks for credit based data and control events on L2CAP.

Parameters

in	*pCallback*	Callback function for data plane messages
in	*pCtrlCallback*	Callback function for control plane messages

Returns

Result of the operation

8.3.2 bleResult_t L2ca_RegisterLePsm (uint16_t *lePsm,* uint16_t *lePsmMtu*)

Registers the LE_PSM from the L2CAP.

Parameters

in	*lePsm*	Bluetooth SIG or Vendor LE_PSM
in	*lePsmMtu*	MTU of the registered PSM

Returns

Result of the operation

8.3.3 bleResult_t L2ca_DeregisterLePsm (uint16_t *lePsm*)

Unregisters the LE_PSM from the L2CAP.

Parameters

in	*lePsm*	Bluetooth SIG or Vendor LE_PSM

Returns

Result of the operation

Precondition

A LE_PSM must be registered a priori

8.3.4 bleResult_t L2ca_ConnectLePsm (uint16_t *lePsm,* deviceId_t *deviceId,* uint16_t *initialCredits*)

Initiates a connection with a peer device for a registered LE_PSM.

Parameters

in	*lePsm*	Bluetooth SIG or Vendor LE_PSM
in	*deviceId*	The DeviceID for which the command is intended
in	*initialCredits*	Initial credits

Returns

Result of the operation

Precondition

A LE_PSM must be registered a priori

8.3.5 bleResult_t L2ca_DisconnectLeCbChannel (deviceId_t *deviceId,* uint16_t *channelId*)

Disconnects a peer device for a registered LE_PSM.

Parameters

in	*deviceId*	The DeviceID for which the command is intended
in	*channelId*	The L2CAP Channel Id assigned on the initiator

Returns

Result of the operation

Precondition

A connection must have already been created

Remarks

Once this command is issued, all incoming data in transit for this device shall be discarded and any new additional outgoing data shall be discarded.

8.3.6 bleResult_t L2ca_CancelConnection (uint16_t *lePsm,* deviceId_t *deviceId,* l2caLeCbConnectionRequestResult_t *refuseReason*)

Terminates an L2CAP channel.

Parameters

in	*lePsm*	Bluetooth SIG or Vendor LE_PSM
in	*deviceId*	The DeviceID for which the command is intended
in	*refuseReason*	Reason to refuse the channel creation

Returns

Result of the operation

Remarks

This interface can be used for a connection pending creation.

8.3.7 bleResult_t L2ca_SendLeCbData (deviceId_t *deviceId,* uint16_t *channelId,* const uint8_t ∗ *pPacket,* uint16_t *packetLength*)

Sends a data packet through a Credit-Based Channel.

Parameters

in	*deviceId*	The DeviceID for which the command is intended
in	*channelId*	The L2CAP Channel Id assigned on the initiator
in	*pPacket*	Data buffer to be transmitted
in	*packetLength*	Length of the data buffer

Returns

Result of the operation

Precondition

An L2CAP Credit Based connection must be in place

8.3.8 bleResult_t L2ca_SendLeCredit (deviceId_t *deviceId,* uint16_t *channelId,* uint16_t *credits*)

Sends credits to a device when capable of receiving additional LE-frames

Parameters

in	*deviceId*	The DeviceID to which credits are given
in	*channelId*	The L2CAP Channel Id assigned on the initiator
in	*credits*	Number of credits to be given

Returns

Result of the operation

Precondition

An L2CAP Credit Based connection must be in place