DM API

Generated by Doxygen 1.8.13

Contents

1 Module Documentation **1**

1.1 GAP Device Manager (DM) . . . 1

1.1.1 Detailed Description . . . 1

1.1.2 Introduction . . . 1

1.1.3 DM Advertising . . . 2

1.1.3.1 Introduction . . . 2

1.1.4 DM Scanning and Discovery . . . 2

1.1.4.1 Introduction . . . 2

1.1.4.2 DM Synchronization Behavior . . . 2

1.1.5 DM Connections . . . 2

1.1.5.1 Introduction . . . 2

1.1.5.2 DM Connection Behavior . . . 2

1.1.6 DM Local Device Management . . . 2

1.1.7 DM Security . . . 3

1.1.8 DM Privacy . . . 3

1.1.9 DM PHY . . . 3

1.1.10 Usage Scenarios . . . 3

1.1.10.1 Advertising and Scanning . . . 3

1.1.10.2 Connection Open and Close . . . 3

1.1.10.3 Pairing . . . 3

1.1.10.4 Encryption . . . 4

1.1.10.5 Privacy . . . 4

1.1.10.6 ECC Key Generation . . . 4

1.1.10.7 Out of Band Confirm Calculation . . . 4
1.2 Device Manager API . . . 6
1.2.1 Detailed Description . . . 31
1.2.2 Macro Definition Documentation . . . 31
1.2.2.1 DM_RAND_ADDR_SA . . . 31
1.2.2.2 DM_RAND_ADDR_RPA . . . 32
1.2.3 Enumeration Type Documentation . . . 32
1.2.3.1 anonymous enum . . . 32
1.2.3.2 anonymous enum . . . 32
1.2.4 Function Documentation . . . 36
1.2.4.1 DmRegister() . . . 36
1.2.4.2 DmFindAdType() . . . 37
1.2.4.3 DmAdvInit() . . . 37
1.2.4.4 DmExtAdvInit() . . . 37
1.2.4.5 DmAdvModeLeg() . . . 38
1.2.4.6 DmAdvModeExt() . . . 38
1.2.4.7 DmAdvConfig() . . . 38
1.2.4.8 DmAdvSetData() . . . 39
1.2.4.9 DmAdvStart() . . . 39
1.2.4.10 DmAdvStop() . . . 40
1.2.4.11 DmAdvRemoveAdvSet() . . . 40
1.2.4.12 DmAdvClearAdvSets() . . . 40
1.2.4.13 DmAdvSetRandAddr() . . . 41
1.2.4.14 DmAdvSetInterval() . . . 41
1.2.4.15 DmAdvSetChannelMap() . . . 41
1.2.4.16 DmAdvSetAddrType() . . . 42
1.2.4.17 DmAdvSetAdValue() . . . 42
1.2.4.18 DmAdvSetName() . . . 43
1.2.4.19 DmDevPrivInit() . . . 43
1.2.4.20 DmDevPrivStart() . . . 43

1.2.4.21 DmDevPrivStop() . . . 44
1.2.4.22 DmAdvUseLegacyPdu() . . . 44
1.2.4.23 DmAdvOmitAdvAddr() . . . 44
1.2.4.24 DmAdvIncTxPwr() . . . 45
1.2.4.25 DmAdvSetPhyParam() . . . 45
1.2.4.26 DmAdvScanReqNotifEnable() . . . 46
1.2.4.27 DmAdvSetFragPref() . . . 46
1.2.4.28 DmAdvSetSid() . . . 46
1.2.4.29 DmPerAdvConfig() . . . 47
1.2.4.30 DmPerAdvSetData() . . . 47
1.2.4.31 DmPerAdvStart() . . . 48
1.2.4.32 DmPerAdvStop() . . . 48
1.2.4.33 DmPerAdvSetInterval() . . . 48
1.2.4.34 DmPerAdvIncTxPwr() . . . 49
1.2.4.35 DmPerAdvEnabled() . . . 49
1.2.4.36 DmExtMaxAdvDataLen() . . . 49
1.2.4.37 DmPrivInit() . . . 50
1.2.4.38 DmPrivResolveAddr() . . . 50
1.2.4.39 DmPrivAddDevToResList() . . . 51
1.2.4.40 DmPrivRemDevFromResList() . . . 51
1.2.4.41 DmPrivClearResList() . . . 52
1.2.4.42 DmPrivReadPeerResolvableAddr() . . . 52
1.2.4.43 DmPrivReadLocalResolvableAddr() . . . 53
1.2.4.44 DmPrivSetAddrResEnable() . . . 53
1.2.4.45 DmPrivSetResolvablePrivateAddrTimeout() . . . 54
1.2.4.46 DmPrivSetPrivacyMode() . . . 54
1.2.4.47 DmPrivGenerateAddr() . . . 55
1.2.4.48 DmLlPrivEnabled() . . . 55
1.2.4.49 DmScanInit() . . . 55
1.2.4.50 DmExtScanInit() . . . 56

1.2.4.51 DmPastInit() . . . 56
1.2.4.52 DmConnCteInit() . . . 56
1.2.4.53 DmScanModeLeg() . . . 56
1.2.4.54 DmScanModeExt() . . . 57
1.2.4.55 DmScanStart() . . . 57
1.2.4.56 DmScanStop() . . . 57
1.2.4.57 DmScanSetInterval() . . . 58
1.2.4.58 DmScanSetAddrType() . . . 58
1.2.4.59 DmSyncStart() . . . 58
1.2.4.60 DmSyncStop() . . . 59
1.2.4.61 DmSyncSetEncrypt() . . . 59
1.2.4.62 DmSyncEncrypted() . . . 60
1.2.4.63 DmSyncEnabled() . . . 60
1.2.4.64 DmSyncInitialRptEnable() . . . 60
1.2.4.65 DmBigSyncStart() . . . 61
1.2.4.66 DmBigSyncStop() . . . 61
1.2.4.67 DmBisSyncInUse() . . . 62
1.2.4.68 DmBigSyncSetBcastCode() . . . 62
1.2.4.69 DmBigSyncSetSecLevel() . . . 63
1.2.4.70 DmBigSyncGetSecLevel() . . . 63
1.2.4.71 DmBisMasterInit() . . . 63
1.2.4.72 DmAddDeviceToPerAdvList() . . . 64
1.2.4.73 DmRemoveDeviceFromPerAdvList() . . . 64
1.2.4.74 DmClearPerAdvList() . . . 64
1.2.4.75 DmPastRptRcvEnable() . . . 65
1.2.4.76 DmPastSyncTrsf() . . . 65
1.2.4.77 DmPastSetInfoTrsf() . . . 65
1.2.4.78 DmPastConfig() . . . 66
1.2.4.79 DmPastDefaultConfig() . . . 66
1.2.4.80 DmConnCteRxSampleStart() . . . 67

1.2.4.81 DmConnCteRxSampleStop() . . . 67
1.2.4.82 DmConnCteTxConfig() . . . 68
1.2.4.83 DmConnCteReqStart() . . . 68
1.2.4.84 DmConnCteReqStop() . . . 69
1.2.4.85 DmConnCteRspStart() . . . 69
1.2.4.86 DmConnCteRspStop() . . . 69
1.2.4.87 DmConnCteGetReqState() . . . 70
1.2.4.88 DmConnCteGetRspState() . . . 70
1.2.4.89 DmReadAntennaInfo() . . . 71
1.2.4.90 DmConnInit() . . . 71
1.2.4.91 DmConnMasterInit() . . . 71
1.2.4.92 DmExtConnMasterInit() . . . 72
1.2.4.93 DmConnSlaveInit() . . . 72
1.2.4.94 DmExtConnSlaveInit() . . . 72
1.2.4.95 DmConnRegister() . . . 72
1.2.4.96 DmConnOpen() . . . 73
1.2.4.97 DmConnClose() . . . 73
1.2.4.98 DmConnAccept() . . . 74
1.2.4.99 DmConnUpdate() . . . 74
1.2.4.100 DmConnSetScanInterval() . . . 75
1.2.4.101 DmExtConnSetScanInterval() . . . 75
1.2.4.102 DmConnSetConnSpec() . . . 75
1.2.4.103 DmExtConnSetConnSpec() . . . 77
1.2.4.104 DmConnSetAddrType() . . . 77
1.2.4.105 DmConnSetIdle() . . . 78
1.2.4.106 DmConnCheckIdle() . . . 78
1.2.4.107 DmConnReadRssi() . . . 78
1.2.4.108 DmRemoteConnParamReqReply() . . . 79
1.2.4.109 DmRemoteConnParamReqNegReply() . . . 79
1.2.4.110 DmConnSetDataLen() . . . 79

1.2.4.111 DmConnRole() . . . 80
1.2.4.112 DmWriteAuthPayloadTimeout() . . . 80
1.2.4.113 DmConnRequestPeerSca() . . . 81
1.2.4.114 DmCisInit() . . . 81
1.2.4.115 DmCisMasterInit() . . . 81
1.2.4.116 DmCisSlaveInit() . . . 82
1.2.4.117 DmCisCigSetSduInterval() . . . 82
1.2.4.118 DmCisCigSetSca() . . . 82
1.2.4.119 DmCisCigSetPackingFraming() . . . 83
1.2.4.120 DmCisCigSetTransLatInterval() . . . 83
1.2.4.121 DmCisCigConfig() . . . 84
1.2.4.122 DmCisCigRemove() . . . 84
1.2.4.123 DmCisOpen() . . . 85
1.2.4.124 DmCisAccept() . . . 85
1.2.4.125 DmCisReject() . . . 85
1.2.4.126 DmCisClose() . . . 86
1.2.4.127 DmCisIdByHandle() . . . 86
1.2.4.128 DmCisHandleById() . . . 87
1.2.4.129 DmCisConnInUse() . . . 87
1.2.4.130 DmCisConnRole() . . . 87
1.2.4.131 DmCisCigInUse() . . . 88
1.2.4.132 DmCisInUse() . . . 88
1.2.4.133 DmBisSlaveInit() . . . 88
1.2.4.134 DmBigStart() . . . 89
1.2.4.135 DmBigStop() . . . 89
1.2.4.136 DmBisInUse() . . . 90
1.2.4.137 DmBigSetPhy() . . . 90
1.2.4.138 DmBigSetPackingFraming() . . . 90
1.2.4.139 DmBigSetBcastCode() . . . 91
1.2.4.140 DmBigSetSecLevel() . . . 91

1.2.4.141 DmBigGetSecLevel() . . . 92
1.2.4.142 DmIsoInit() . . . 92
1.2.4.143 DmIsoRegister() . . . 92
1.2.4.144 DmIsoDataPathSetup() . . . 93
1.2.4.145 DmIsoDataPathRemove() . . . 93
1.2.4.146 DmDataPathConfig() . . . 93
1.2.4.147 DmReadLocalSupCodecs() . . . 94
1.2.4.148 DmReadLocalSupCodecCap() . . . 94
1.2.4.149 DmReadLocalSupCtrDly() . . . 94
1.2.4.150 DmSendIsoData() . . . 95
1.2.4.151 DmSetDefaultPhy() . . . 95
1.2.4.152 DmReadPhy() . . . 96
1.2.4.153 DmSetPhy() . . . 96
1.2.4.154 DmPhyInit() . . . 96
1.2.4.155 DmDevReset() . . . 97
1.2.4.156 DmDevSetRandAddr() . . . 97
1.2.4.157 DmDevWhiteListAdd() . . . 97
1.2.4.158 DmDevWhiteListRemove() . . . 98
1.2.4.159 DmDevWhiteListClear() . . . 98
1.2.4.160 DmDevSetFilterPolicy() . . . 98
1.2.4.161 DmDevSetExtFilterPolicy() . . . 99
1.2.4.162 DmDevVsInit() . . . 99
1.2.4.163 DmSecInit() . . . 100
1.2.4.164 DmSecLescInit() . . . 100
1.2.4.165 DmSecPairReq() . . . 100
1.2.4.166 DmSecPairRsp() . . . 101
1.2.4.167 DmSecCancelReq() . . . 101
1.2.4.168 DmSecAuthRsp() . . . 101
1.2.4.169 DmSecSlaveReq() . . . 103
1.2.4.170 DmSecEncryptReq() . . . 103

1.2.4.171 DmSecLtkRsp() . . . 104
1.2.4.172 DmSecSetLocalCsrk() . . . 104
1.2.4.173 DmSecSetLocalIrk() . . . 105
1.2.4.174 DmSecGenerateEccKeyReq() . . . 105
1.2.4.175 DmSecSetEccKey() . . . 105
1.2.4.176 DmSecGetEccKey() . . . 106
1.2.4.177 DmSecSetDebugEccKey() . . . 106
1.2.4.178 DmSecSetOob() . . . 106
1.2.4.179 DmSecCalcOobReq() . . . 106
1.2.4.180 DmSecCompareRsp() . . . 107
1.2.4.181 DmSecGetCompareValue() . . . 107
1.2.4.182 DmLlAddrType() . . . 108
1.2.4.183 DmHostAddrType() . . . 108
1.2.4.184 DmSizeOfEvt() . . . 108
1.2.4.185 DmL2cConnUpdateCnf() . . . 109
1.2.4.186 DmL2cCmdRejInd() . . . 109
1.2.4.187 DmL2cConnUpdateInd() . . . 110
1.2.4.188 DmConnIdByHandle() . . . 110
1.2.4.189 DmConnInUse() . . . 110
1.2.4.190 DmConnActiveCount() . . . 111
1.2.4.191 DmConnPeerAddrType() . . . 111
1.2.4.192 DmConnPeerAddr() . . . 111
1.2.4.193 DmConnLocalAddrType() . . . 112
1.2.4.194 DmConnLocalAddr() . . . 112
1.2.4.195 DmConnPeerRpa() . . . 112
1.2.4.196 DmConnLocalRpa() . . . 113
1.2.4.197 DmConnSecLevel() . . . 113
1.2.4.198 DmSmpEncryptReq() . . . 114
1.2.4.199 DmSmpCbackExec() . . . 114
1.2.4.200 DmSecGetLocalCsrk() . . . 114

1.2.4.201 DmSecGetLocalIrk() . . . 115
1.2.4.202 DmReadRemoteFeatures() . . . 115
1.2.4.203 DmReadRemoteVerInfo() . . . 115
1.2.4.204 DmDisableSlaveLatency() . . . 116
1.2.4.205 DmOverruleRemoteMaxRxOctetsAndTime() . . . 116
1.2.4.206 HciVsdSetDeviceAddress() . . . 116
1.2.4.207 HciVsdSetTransmitPower() . . . 117
1.2.4.208 HciCmndVsdSetLeMetaVSDEvent() . . . 117
1.2.4.209 HciCmndVsdResetLeMetaVSDEvent() . . . 117
1.3 STACK_EVENT . . . 119
1.3.1 Detailed Description . . . 119
1.3.2 Function Documentation . . . 119
1.3.2.1 DmHandlerInit() . . . 119
1.3.2.2 DmHandler() . . . 119
1.4 WSF_OS_API . . . 121
1.4.1 Detailed Description . . . 122
1.4.2 Typedef Documentation . . . 122
1.4.2.1 wsfEventHandler_t . . . 122
1.4.3 Function Documentation . . . 123
1.4.3.1 WsfSetEvent() . . . 123
1.4.3.2 WsfTaskSetReady() . . . 123
1.4.3.3 WsfTaskMsgQueue() . . . 123
1.4.3.4 WsfOsSetNextHandler() . . . 124
1.4.3.5 WsfOsInit() . . . 124
1.4.3.6 WsfOsReadyToSleep() . . . 124
1.4.3.7 WsfOsDispatcher() . . . 125
1.4.3.8 WsfOsRegisterIdleTask() . . . 125
1.5 WSF_TYPES . . . 126
1.5.1 Detailed Description . . . 126

2 Data Structure Documentation **127**

2.1 dmAdvNewAddrIndEvt_t Struct Reference . . . 127

2.1.1 Detailed Description . . . 128

2.2 dmAdvSetStartEvt_t Struct Reference . . . 128

2.2.1 Detailed Description . . . 129

2.3 dmCfg_t Struct Reference . . . 129

2.3.1 Detailed Description . . . 129

2.4 dmEvt_t Union Reference . . . 130

2.4.1 Detailed Description . . . 133

2.5 dmL2cCmdRejEvt_t Struct Reference . . . 134

2.5.1 Detailed Description . . . 134

2.6 dmPerAdvSetStartEvt_t Struct Reference . . . 135

2.6.1 Detailed Description . . . 135

2.7 dmPerAdvSetStopEvt_t Struct Reference . . . 136

2.7.1 Detailed Description . . . 136

2.8 dmPrivGenAddrIndEvt_t Struct Reference . . . 137

2.8.1 Detailed Description . . . 137

2.9 dmRemoveIsoDataPathEvt_t Struct Reference . . . 138

2.9.1 Detailed Description . . . 138

2.10 dmSecAuthReqIndEvt_t Struct Reference . . . 139

2.10.1 Detailed Description . . . 139

2.11 dmSecCnfIndEvt_t Struct Reference . . . 140

2.11.1 Detailed Description . . . 140

2.12 dmSecCsrk_t Struct Reference . . . 141

2.12.1 Detailed Description . . . 141

2.13 dmSecEncryptIndEvt_t Struct Reference . . . 141

2.13.1 Detailed Description . . . 142

2.14 dmSecIrk_t Struct Reference . . . 142

2.14.1 Detailed Description . . . 143

2.15 dmSecKey_t Union Reference . . . 143

2.15.1 Detailed Description . . . 144
2.16 dmSecKeyIndEvt_t Struct Reference . . . 144
2.16.1 Detailed Description . . . 145
2.17 dmSecKeypressIndEvt_t Struct Reference . . . 146
2.17.1 Detailed Description . . . 146
2.18 dmSecLescOobCfg_t Struct Reference . . . 147
2.18.1 Detailed Description . . . 147
2.19 dmSecLtk_t Struct Reference . . . 148
2.19.1 Detailed Description . . . 148
2.20 dmSecOobCalcIndEvt_t Struct Reference . . . 149
2.20.1 Detailed Description . . . 149
2.21 dmSecPairCmplIndEvt_t Struct Reference . . . 150
2.21.1 Detailed Description . . . 150
2.22 dmSecPairIndEvt_t Struct Reference . . . 151
2.22.1 Detailed Description . . . 151
2.23 dmSecSlaveIndEvt_t Struct Reference . . . 152
2.23.1 Detailed Description . . . 152
2.24 dmSetupIsoDataPathEvt_t Struct Reference . . . 153
2.24.1 Detailed Description . . . 153
2.25 wsfMsgHdr_t Struct Reference . . . 154
2.25.1 Detailed Description . . . 154

3 File Documentation **155**

3.1 /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h File Reference . . . 155
3.1.1 Detailed Description . . . 177
3.2 /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_handler.h File Reference . . . 177
3.2.1 Detailed Description . . . 178
3.3 /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/wsf/include/wsf_os.h File Reference . . . 179
3.3.1 Detailed Description . . . 181
3.4 /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/wsf/include/wsf_types.h File Reference . . . 182
3.4.1 Detailed Description . . . 183

Index **185**

Chapter 1

Module Documentation

1.1 GAP Device Manager (DM)

Modules

- Device Manager API

1.1.1 Detailed Description

1.1.2 Introduction

The DM subsystem implements device management procedures required by the stack. These procedures are partitioned by procedure category and device role (master or slave). The following procedures are implemented in DM:

- Advertising and device visibility: Enable/disable advertising, set advertising parameters and data, set connectability and discoverability.
- Scanning and device discovery: Start/stop scanning, set scan parameters, advertising reports, name discovery.
- Connection management: Create/accept/remove connections, set/update connection parameters, read R↩ SSI.
- Local device management: Initialization and reset, set local parameters, vendor-specific commands, LE GAP attribute management.
- Security management: Bonding, storage of security parameters, authentication, encryption, authorization, random address management.
- Privacy management: maintenance of resolving list, random address usage and privacy mode.
- PHY control

DM procedures are implemented in support of the GAP profile when applicable.

The Device Manager controls all GAP behavior which includes the following:

- DM Advertising
- DM Scanning and Discovery
- DM Connections
- DM Privacy

For full API, see Device Manager API

1.1.3 DM Advertising

1.1.3.1 Introduction

The DM interface for advertising and device visibility configures, enables, and disables the advertising procedure. A device advertises when it wants to connect to or be discovered by other devices. Devices may also advertise to simply broadcast data.

This interface can only be used when operating as a slave or broadcaster.

This implementation handles Legacy and Extended Advertising including Periodic Advertising.

1.1.4 DM Scanning and Discovery

1.1.4.1 Introduction

The DM scanning and device discovery interface configures, enables, and disables the scanning procedure. A device scans when it wants to discover or connect to other devices. A device may also scan simply to receive broadcast advertisements.

This interface can only be used when operating as a master or observer.

This implementation handles Legacy and Extended Scanning features including Synchronization on Periodic Advertisements.

1.1.4.2 DM Synchronization Behavior

Here is the state machine for DM_Synchronization.

1.1.5 DM Connections

1.1.5.1 Introduction

The DM connection management interface is used to open, accept, configure, and close connections. It is also used to read connection-related information such as the RSSI, channel map, and remote device information.

1.1.5.2 DM Connection Behavior

Here is the state machine for DM_Connections

1.1.6 DM Local Device Management

The DM local device management interface is used for initialization and reset, setting local parameters, sending vendor-specific commands, and LE GAP attribute management.

1.1.7 DM Security

The DM security management interface is used for pairing, authentication, and encryption.

1.1.8 DM Privacy

The DM Privacy interface is used by a master or slave device for private address resolution.

1.1.9 DM PHY

The DM PHY interface is used by a master or slave device to access the transmitter and receiver PHY settings for connections.

1.1.10 Usage Scenarios

1.1.10.1 Advertising and Scanning

Figure 2 shows a master device performing a scan and a slave device advertising. The slave application first configures the advertising parameters by calling DmAdvSetInterval() to set the advertising interval and then DmAdvSetData() twice to set the advertising data and the scan response data. Then it calls DmAdvStart() to start advertising.

The master application configures the scan interval and then calls DmScanStart() to begin scanning. When advertisements are received the stack sends DM_SCAN_REPORT_IND events to the application. The master application stops scanning by calling DmScanStop(). The slave application stops advertising by calling DmAdvStop().

1.1.10.2 Connection Open and Close

Figure 3 shows connection procedures between two devices. The scenario starts with the slave device advertising and the master device already having the address of the slave. The master application calls DmConnOpen() to initiate a connection. A connection is established and a DM_CONN_OPEN_IND is sent to the application from the stack on each device.

Next, the master performs a connection update by calling DmConnUpdate(). When the connection update is complete, a DM_CONN_UPDATE_IND is sent to the application from the stack on each device.

Next, the slave closes the connection by calling DmConnClose(). A DM_CONN_CLOSE_IND event is sent from the stack on each device when the connection is closed.

1.1.10.3 Pairing

Figure 4 shows a pairing procedure between two devices. A connection is established between the two devices and the master application initiates pairing by calling DmSecPairReq(). The slave application receives a DM_SEC_PAIR_IND and calls DmSecPairRsp() to proceed with pairing. In this example, a PIN is used and a DM_SEC_AUTH_REQ_IND is sent to the application on each device to request a PIN. Each application responds with the PIN by calling DmSecAuthRsp().

In the next phase of pairing, the connection is encrypted and a DM_SEC_ENCRYPT_IND event is sent to the application on each device. Then key exchange begins. According to the Bluetooth specification, the slave device always distributes keys first. In this example, the slave distributes two keys and the master device distributes one. The slave sends its key data to the master. Note that when the slave sends its LTK, the slave application receives a DM_SEC_KEY_IND containing its own LTK. Then the master sends its key data to the slave. When the key exchange is completed successfully, a DM_SEC_PAIR_CMPL_IND event is sent to the application on each device.

1.1.10.4 Encryption

Figure 5 shows an encryption procedure. In this example the slave device requests security by calling DmSecSlaveReq() to sends a slave security request message to the master. The stack on the master sends a DM_SEC_SLAVE_REQ_IND to the application. On receiving the event the master application determines that this is a bonded device and its LTK is available, so it calls DmSecEncryptReq() to enable encryption.

After the encryption procedure is initiated the slave application receives a DM_SEC_LTK_REQ_IND, requesting the LTK used with this master device. The application finds the key and calls DmSecLtkRsp(). The encryption procedure completes and a DM_SEC_ENCRYPT_IND event is sent to the application on each device.

1.1.10.5 Privacy

Figure 6 shows a master device performing a scan and a slave device advertising with a private resolvable address. Before a master device can resolve a slave's address, the devices must have paired and the master must have received the slave's IRK during pairing.

The slave application first enables use of a private resolvable address by calling DmDevPrivStart(). If this is the first time since device reset that DmDevPrivStart() has been called, the application must wait for a DM_ADV_NEW_ADDR_IND before it starts advertising. Then it calls DmAdvStart() to start advertising.

The master application calls DmScanStart() to begin scanning. When advertisements are received the stack sends DM_SCAN_REPORT_IND events to the application. The master application calls DmPrivResolveAddr() with the address and address type from the scan report to resolve the address with the IRK it had received previously. After the slave application stops advertising, it can call DmDevPrivStop() to stop using a private resolvable address.

1.1.10.6 ECC Key Generation

An ECC Key must be stored in the Device Manager before use in LE Secure Connections pairing. The Device Manager can generate an ECC, Elliptic Curve Cryptography, key, or the application can store an ECC Key in Non-Volatile storage. An ECC key cannot be generated until after the Device Manager reset is complete.

To generate an ECC Key, call the DmSecGenerateEccKeyReq() function after receiving the DM_RESET_CMPL_IND event. The DM_SEC_ECC_KEY_IND event will be called after the ECC Key generation is complete. The ECC Key can then be stored into the DM using the DmSecSetEccKey() function.

Note: For some applications, it may be desirable to skip ECC Key Generation and store an ECC key in Non Volatile storage. In these situations, the ECC key can be written to the Device Manager with DmSecSetEccKey() any time after the DM is reset, and before pairing begins.

Note: The Device Manager makes use of the Security service to generate and validate ECC keys. The Security service's ECC subsystem may need to be ported to an application's target hardware or software framework for LE Secure Connections to operate properly.

The following figure shows the ECC Key generation scenario:

1.1.10.7 Out of Band Confirm Calculation

When using Out-of-Band (OOB) LE Secure Connections pairing, devices must generate random and confirm values. Furthermore, the devices must exchange random and confirm values through an out-of-band mechanism. At which point, the local and peer random and confirm values must be stored in the Device Manager before OOB pairing starts .

The OOB confirm calculation can be performed with DmSecCalcOobReq(), and requires an ECC, Elliptic Curve Cryptography, key. Therefore, on receipt of the ECC key indication event, DM_SEC_ECC_KEY_IND, an application may call the DmSecCalcOobReq() function to calculate an OOB confirm value. The result of the confirm calculation is returned via the DM_SEC_CALC_OOB_IND event.

After an application exchanges random and confirm values via an out-of-band mechanism with a peer, the application must store the local random and confirm values in the device manager. This is performed with the DmSecSetOob() function on receipt of the DM_SEC_AUTH_REQ_IND event. The following figure shows the OOB confirm calculation scenario:

1.2 Device Manager API

Data Structures

- struct dmCfg_t
  Configuration structure.
- struct dmSecLtk_t
  LTK data type.
- struct dmSecIrk_t
  IRK data type.
- struct dmSecCsrk_t
  CSRK data type.
- union dmSecKey_t
  Union of key types.
- struct dmSecPairCmplIndEvt_t
  Data type for DM_SEC_PAIR_CMPL_IND.
- struct dmSecEncryptIndEvt_t
  Data type for DM_SEC_ENCRYPT_IND.
- struct dmSecAuthReqIndEvt_t
  Data type for DM_SEC_AUTH_REQ_IND.
- struct dmSecPairIndEvt_t
  Data type for DM_SEC_PAIR_IND.
- struct dmSecSlaveIndEvt_t
  Data type for DM_SEC_SLAVE_REQ_IND.
- struct dmSecKeyIndEvt_t
  Data type for DM_SEC_KEY_IND.
- struct dmSecCnfIndEvt_t
  Data type for DM_SEC_COMPARE_IND.
- struct dmSecKeypressIndEvt_t
  Data type for DM_SEC_KEYPRESS_IND.
- struct dmPrivGenAddrIndEvt_t
  Data type for DM_PRIV_GENERATE_ADDR_IND.
- struct dmSecOobCalcIndEvt_t
  Data type for DM_SEC_CALC_OOB_IND.
- struct dmAdvNewAddrIndEvt_t
  Data type for DM_ADV_NEW_ADDR_IND.
- struct dmAdvSetStartEvt_t
  Data structure for DM_ADV_SET_START_IND.
- struct dmPerAdvSetStartEvt_t
  Data structure for DM_PER_ADV_SET_START_IND.
- struct dmPerAdvSetStopEvt_t
  Data structure for DM_PER_ADV_SET_STOP_IND.
- struct dmSetupIsoDataPathEvt_t
  Data structure for DM_ISO_DATA_PATH_SETUP_IND.
- struct dmRemoveIsoDataPathEvt_t
  Data structure for DM_ISO_DATA_PATH_REMOVE_IND.
- struct dmL2cCmdRejEvt_t
  Data structure for DM_L2C_CMD_REJ_IND.
- union dmEvt_t
  Union of DM callback event data types.
- struct dmSecLescOobCfg_t
  Data type for DmSecSetOob().

Macros

- #define DM_SEC_HCI_ERR_BASE 0x20

  Base value for HCI error status values for DM_SEC_PAIR_CMPL_IND.

Typedefs

- typedef uint8_t dmConnId_t

  Connection identifier.
- typedef uint8_t dmSyncId_t

  Synchronization identifier.
- typedef void(∗ dmCback_t) (dmEvt_t ∗pDmEvt)

  Callback type.

GAP Device Role

Connectable GAP Roles.

- #define DM_ROLE_MASTER HCI_ROLE_MASTER

  Role is master.
- #define DM_ROLE_SLAVE HCI_ROLE_SLAVE

  Role is slave.

GAP Discovery Mode

When setup as a discoverable device, these are the possible modes of discovery.

- #define DM_DISC_MODE_NONE 0

  GAP non-discoverable.
- #define DM_DISC_MODE_LIMITED 1

  GAP limited discoverable mode.
- #define DM_DISC_MODE_GENERAL 2

  GAP general discoverable mode.

GAP Advertising Type

Type of connectable or disconverable advertising to perform.

- #define DM_ADV_CONN_UNDIRECT 0

  Connectable and scannable undirected advertising.
- #define DM_ADV_CONN_DIRECT 1

  Connectable directed advertising.
- #define DM_ADV_SCAN_UNDIRECT 2

  Scannable undirected advertising.
- #define DM_ADV_NONCONN_UNDIRECT 3

  Non-connectable and non-scannable undirected advertising.
- #define DM_ADV_CONN_DIRECT_LO_DUTY 4

  Connectable directed low duty cycle advertising.

GAP AE Advertising Types

Advertising extension types - AE only.

- #define DM_EXT_ADV_CONN_UNDIRECT 5

  Connectable undirected advertising.
- #define DM_EXT_ADV_NONCONN_DIRECT 6

  Non-connectable and non-scannable directed advertising.
- #define DM_EXT_ADV_SCAN_DIRECT 7

  Scannable directed advertising.
- #define DM_ADV_NONE 255

  For internal use only.

GAP Advertising Report Type

Type of an advertising report observed while scanning.

- #define DM_RPT_CONN_UNDIRECT 0

  Connectable and scannable undirected advertising.
- #define DM_RPT_CONN_DIRECT 1

  Connectable directed advertising.
- #define DM_RPT_SCAN_UNDIRECT 2

  Scannable undirected advertising.
- #define DM_RPT_NONCONN_UNDIRECT 3

  Non-connectable undirected advertising.
- #define DM_RPT_SCAN_RESPONSE 4

  Scan response.

GAP Advertising Data Location

Whether data is located in the advertising data or in the scan response data

- #define DM_DATA_LOC_ADV 0

  Locate data in the advertising data.
- #define DM_DATA_LOC_SCAN 1

  Locate data in the scan response data.

GAP Scan Type

When setup as a connectable or observer device, this is the type of scanning to perform.

- #define DM_SCAN_TYPE_PASSIVE 0

  Passive scan.
- #define DM_SCAN_TYPE_ACTIVE 1

  Active scan.

GAP Advertising Channel Map

Advertising channel map codes

- #define DM_ADV_CHAN_37 HCI_ADV_CHAN_37

  Advertising channel 37.
- #define DM_ADV_CHAN_38 HCI_ADV_CHAN_38

  Advertising channel 38.
- #define DM_ADV_CHAN_39 HCI_ADV_CHAN_39

  Advertising channel 39.
- #define DM_ADV_CHAN_ALL (HCI_ADV_CHAN_37 | HCI_ADV_CHAN_38 | HCI_ADV_CHAN_39)

  All advertising channels.

DM Client IDs

The client ID parameter to function DmConnRegister()

- #define DM_CLIENT_ID_ATT 0

  Identifier for attribute protocol, for internal use only.
- #define DM_CLIENT_ID_SMP 1

  Identifier for security manager protocol, for internal use only.
- #define DM_CLIENT_ID_DM 2

  Identifier for device manager, for internal use only.
- #define DM_CLIENT_ID_APP 3

  Identifier for the application.
- #define DM_CLIENT_ID_L2C 4

  Identifier for L2CAP.
- #define DM_CLIENT_ID_MAX 5

  For internal use only.

DM Unknown IDs

Values for unknown or unspecificed device identifiers.

- #define DM_CONN_ID_NONE 0

  Unknown connection ID or other error.
- #define DM_SYNC_ID_NONE 0

  Unknown sync ID or other error.
- #define DM_CIG_ID_NONE 0xFF

  Unknown Connected Isochronous Group (CIG) ID or other error.
- #define DM_CIS_ID_NONE 0xFF

  Unknown Connected Isochronous Stream (CIS) ID or other error.

GAP Address Type

The address type to use over the air or that is associated with a received address.

- #define DM_ADDR_PUBLIC 0x00
  Public device address.
- #define DM_ADDR_RANDOM 0x01
  Random device address.
- #define DM_ADDR_PUBLIC_IDENTITY 0x02
  Public identity address (corresponds to resolved private address)
- #define DM_ADDR_RANDOM_IDENTITY 0x03
  Random (static) identity address (corresponds to resolved private address)
- #define DM_ADDR_RANDOM_UNRESOLVED 0xFE
  Random device address (Controller unable to resolve)
- #define DM_ADDR_NONE 0xFF
  No address provided (anonymous)

GAP Advertising Data Types

Advertising data types flags.

- #define DM_ADV_TYPE_FLAGS 0x01
  Flag bits.
- #define DM_ADV_TYPE_16_UUID_PART 0x02
  Partial list of 16 bit UUIDs.
- #define DM_ADV_TYPE_16_UUID 0x03
  Complete list of 16 bit UUIDs.
- #define DM_ADV_TYPE_32_UUID_PART 0x04
  Partial list of 32 bit UUIDs.
- #define DM_ADV_TYPE_32_UUID 0x05
  Complete list of 32 bit UUIDs.
- #define DM_ADV_TYPE_128_UUID_PART 0x06
  Partial list of 128 bit UUIDs.
- #define DM_ADV_TYPE_128_UUID 0x07
  Complete list of 128 bit UUIDs.
- #define DM_ADV_TYPE_SHORT_NAME 0x08
  Shortened local name.
- #define DM_ADV_TYPE_LOCAL_NAME 0x09
  Complete local name.
- #define DM_ADV_TYPE_TX_POWER 0x0A
  TX power level.
- #define DM_ADV_TYPE_SM_TK_VALUE 0x10
  Security manager TK value.
- #define DM_ADV_TYPE_SM_OOB_FLAGS 0x11
  Security manager OOB flags.
- #define DM_ADV_TYPE_CONN_INTERVAL 0x12
  Slave preferred connection interval.
- #define DM_ADV_TYPE_SIGNED_DATA 0x13
  Signed data.

- #define DM_ADV_TYPE_16_SOLICIT 0x14
  Service soliticiation list of 16 bit UUIDs.
- #define DM_ADV_TYPE_128_SOLICIT 0x15
  Service soliticiation list of 128 bit UUIDs.
- #define DM_ADV_TYPE_SERVICE_DATA 0x16
  Service data - 16-bit UUID.
- #define DM_ADV_TYPE_PUBLIC_TARGET 0x17
  Public target address.
- #define DM_ADV_TYPE_RANDOM_TARGET 0x18
  Random target address.
- #define DM_ADV_TYPE_APPEARANCE 0x19
  Device appearance.
- #define DM_ADV_TYPE_ADV_INTERVAL 0x1A
  Advertising interval.
- #define DM_ADV_TYPE_BD_ADDR 0x1B
  LE Bluetooth device address.
- #define DM_ADV_TYPE_ROLE 0x1C
  LE role.
- #define DM_ADV_TYPE_32_SOLICIT 0x1F
  Service soliticiation list of 32 bit UUIDs.
- #define DM_ADV_TYPE_SVC_DATA_32 0x20
  Service data - 32-bit UUID.
- #define DM_ADV_TYPE_SVC_DATA_128 0x21
  Service data - 128-bit UUID.
- #define DM_ADV_TYPE_LESC_CONFIRM 0x22
  LE Secure Connections confirm value.
- #define DM_ADV_TYPE_LESC_RANDOM 0x23
  LE Secure Connections random value.
- #define DM_ADV_TYPE_URI 0x24
  URI.
- #define DM_ADV_TYPE_INDOOR_POS 0x25
  Indoor positioning service.
- #define DM_ADV_TYPE_TRANS_DISC 0x26
  Transport discovery service.
- #define DM_ADV_TYPE_LE_SUP_FEAT 0x27
  LE supported features.
- #define DM_ADV_TYPE_CH_MAP_UPD_IND 0x28
  Channel map update indication.
- #define DM_ADV_TYPE_PB_ADV 0x29
  PB-ADV.
- #define DM_ADV_TYPE_MESH_MSG 0x2A
  Mesh message.
- #define DM_ADV_TYPE_MESH_BEACON 0x2B
  Mesh beacon.
- #define DM_ADV_TYPE_BIG_INFO 0x2C
  BIG Info.
- #define DM_ADV_TYPE_BCAST_CODE 0x2D
  Mesh beacon.
- #define DM_ADV_TYPE_3D_INFO_DATA 0x3D
  3D information data
- #define DM_ADV_TYPE_MANUFACTURER 0xFF
  Manufacturer specific data.

GAP Advertising Data Flag Advertising Type

Bit mask for Advertising Type flag in advertising data.

- #define DM_FLAG_LE_LIMITED_DISC 0x01
  Limited discoverable flag.
- #define DM_FLAG_LE_GENERAL_DISC 0x02
  General discoverable flag.
- #define DM_FLAG_LE_BREDR_NOT_SUP 0x04
  BR/EDR not supported flag.

GAP Advertising Data Element Indexes

Advertising data element indexes.

- #define DM_AD_LEN_IDX 0
  Advertising data element len.
- #define DM_AD_TYPE_IDX 1
  Advertising data element type.
- #define DM_AD_DATA_IDX 2
  Advertising data element data.

GAP Advertising URI

Advertising URI Scheme

- #define DM_URI_SCHEME_HTTP 0x16
  URI HTTP Scheme.
- #define DM_URI_SCHEME_HTTPS 0x17
  URI HTTPS Scheme.

GAP Timeouts

Timeouts defined by the GAP specification; in units of milliseconds.

- #define DM_GAP_LIM_ADV_TIMEOUT 180000
  Maximum advertising duration in limited discoverable mode.
- #define DM_GAP_GEN_DISC_SCAN_MIN 10240
  Minimum scan duration for general discovery.
- #define DM_GAP_LIM_DISC_SCAN_MIN 10240
  Minimum scan duration for limited discovery.
- #define DM_GAP_CONN_PARAM_TIMEOUT 30000
  Connection parameter update timeout.
- #define DM_GAP_SCAN_FAST_PERIOD 30720
  Minimum time to perform scanning when user initiated.
- #define DM_GAP_ADV_FAST_PERIOD 30000
  Minimum time to perform advertising when user initiated.

GAP 1M PHY Timing

Advertising, scanning, and connection parameters defined in the GAP specification for the LE 1M PHY. In units of 625 microseconds.

- #define DM_GAP_SCAN_FAST_INT_MIN 48
  Minimum scan interval when user initiated.
- #define DM_GAP_SCAN_FAST_INT_MAX 96
  Maximum scan interval when user initiated.
- #define DM_GAP_SCAN_FAST_WINDOW 48
  Scan window when user initiated.
- #define DM_GAP_SCAN_SLOW_INT_1 2048
  Scan interval 1 when background scannning.
- #define DM_GAP_SCAN_SLOW_WINDOW_1 18
  Scan window 1 when background scanning.
- #define DM_GAP_SCAN_SLOW_INT_2 4096
  Scan interval 2 when background scannning.
- #define DM_GAP_SCAN_SLOW_WINDOW_2 36
  Scan window 2 when background scanning.
- #define DM_GAP_ADV_FAST_INT_MIN_1 48
  Minimum advertising interval 1 when user initiated.
- #define DM_GAP_ADV_FAST_INT_MAX_1 96
  Maximum advertising interval 1 when user initiated.
- #define DM_GAP_ADV_FAST_INT_MIN_2 160
  Minimum advertising interval 2 when user initiated.
- #define DM_GAP_ADV_FAST_INT_MAX_2 240
  Maximum advertising interval 2 when user initiated.
- #define DM_GAP_ADV_SLOW_INT_MIN 1600
  Minimum advertising interval when background advertising.
- #define DM_GAP_ADV_SLOW_INT_MAX 1920
  Maximum advertising interval when background advertising.

GAP Coded PHY Timing

Advertising, scanning, and connection parameters defined in the GAP specification for the LE Coded PHY. In units of 625 microseconds.

- #define DM_GAP_SCAN_CODED_FAST_INT_MIN 144
  Minimum scan interval when user initiated on LE Coded PHY.
- #define DM_GAP_SCAN_CODED_FAST_INT_MAX 288
  Maximum scan interval when user initiated on LE Coded PHY.
- #define DM_GAP_SCAN_CODED_FAST_WINDOW 144
  Scan window when user initiated on LE Coded PHY.
- #define DM_GAP_SCAN_CODED_SLOW_INT_1 6144
  Scan interval 1 when background scannning on LE Coded PHY.
- #define DM_GAP_SCAN_CODED_SLOW_WINDOW_1 54
  Scan window 1 when background scanning on LE Coded PHY.
- #define DM_GAP_SCAN_CODED_SLOW_INT_2 12288
  Scan interval 2 when background scannning on LE Coded PHY.

- #define DM_GAP_SCAN_CODED_SLOW_WINDOW_2 108

  Scan window 2 when background scanning on LE Coded PHY.
- #define DM_GAP_ADV_CODED_FAST_INT_MIN_1 144

  Minimum advertising interval 1 when user initiated on LE Coded PHY.
- #define DM_GAP_ADV_CODED_FAST_INT_MAX_1 288

  Maximum advertising interval 1 when user initiated on LE Coded PHY.
- #define DM_GAP_ADV_CODED_FAST_INT_MIN_2 480

  Minimum advertising interval 2 when user initiated on LE Coded PHY.
- #define DM_GAP_ADV_CODED_FAST_INT_MAX_2 720

  Maximum advertising interval 2 when user initiated on LE Coded PHY.
- #define DM_GAP_ADV_CODED_SLOW_INT_MIN 4800

  Minimum advertising interval when background advertising on LE Coded PHY.
- #define DM_GAP_ADV_CODED_SLOW_INT_MAX 5760

  Maximum advertising interval when background advertising on LE Coded PHY.

GAP Connection Slave Latency

- #define DM_GAP_CONN_EST_LATENCY 0

  GAP connection establishment slaves latency.

GAP Connection Interval

GAP connection interval in 1.25ms units.

- #define DM_GAP_INITIAL_CONN_INT_MIN 24

  Minimum initial connection interval.
- #define DM_GAP_INITIAL_CONN_INT_MAX 40

  Maximum initial connection interval.

GAP Connection Event Lengths

GAP connection establishment minimum and maximum connection event lengths.

- #define DM_GAP_CONN_EST_MIN_CE_LEN 0

  Connection establishment minimum event length.
- #define DM_GAP_CONN_EST_MAX_CE_LEN 0

  Connection establishment maximum event length.

GAP Peripheral Privacy Characteristic Values

- #define DM_GAP_PRIV_DISABLED 0

  Privacy Disabled.
- #define DM_GAP_PRIV_ENABLED 1

  Privacy Enabled.

GAP Connection Supervision Timeout

Connection supervision timeout, in 10ms units

- #define DM_DEFAULT_EST_SUP_TIMEOUT 2000
  Connection establishment supervision timeout default, in 10ms units.

GAP Security Pairing Authentication Requirements

Pairing authentication/security properties bit mask.

- #define DM_AUTH_BOND_FLAG SMP_AUTH_BOND_FLAG
  Bonding requested.
- #define DM_AUTH_MITM_FLAG SMP_AUTH_MITM_FLAG
  MITM (authenticated pairing) requested.
- #define DM_AUTH_SC_FLAG SMP_AUTH_SC_FLAG
  LE Secure Connections requested.
- #define DM_AUTH_KP_FLAG SMP_AUTH_KP_FLAG
  Keypress notifications requested.

GAP Key Distribution Flags

Key distribution bit mask

- #define DM_KEY_DIST_LTK SMP_KEY_DIST_ENC
  Distribute LTK used for encryption.
- #define DM_KEY_DIST_IRK SMP_KEY_DIST_ID
  Distribute IRK used for privacy.
- #define DM_KEY_DIST_CSRK SMP_KEY_DIST_SIGN
  Distribute CSRK used for signed data.

DM Security Key Indication Types

Type of key used in DM_SEC_KEY_IND.

- #define DM_KEY_LOCAL_LTK 0x01
  LTK generated locally for this device.
- #define DM_KEY_PEER_LTK 0x02
  LTK received from peer device.
- #define DM_KEY_IRK 0x04
  IRK and identity info of peer device.
- #define DM_KEY_CSRK 0x08
  CSRK of peer device.

GAP Security Level

GAP Mode 1 Security Levels

- #define DM_SEC_LEVEL_NONE 0
  Connection has no security.
- #define DM_SEC_LEVEL_ENC 1
  Connection is encrypted with unauthenticated key.
- #define DM_SEC_LEVEL_ENC_AUTH 2
  Connection is encrypted with authenticated key.
- #define DM_SEC_LEVEL_ENC_LESC 3
  Connection is encrypted with LE Secure Connections.

GAP Broadcast Security Level

GAP Mode 3 Security Levels

- #define DM_SEC_LEVEL_BCAST_NONE 0
  No security (no authentication and no encryption)
- #define DM_SEC_LEVEL_BCAST_UNAUTH 1
  Use of unauthenticated Broadcast_Code.
- #define DM_SEC_LEVEL_BCAST_AUTH 2
  Use of authenticated Broadcast_Code.

GAP Random Address Types

Random address type masks.

- #define DM_RAND_ADDR_STATIC 0xC0
  Static address.
- #define DM_RAND_ADDR_RESOLV 0x40
  Resolvable private address.
- #define DM_RAND_ADDR_NONRESOLV 0x00
  Non-resolvable private address.

GAP Random Address Macros

Macros for identifying address type.

- #define DM_RAND_ADDR_GET(addr) ((addr)[5] & 0xC0)
  Get the type of random address.
- #define DM_RAND_ADDR_SET(addr, type) {(addr)[5] = ((addr)[5] & 0x3F) | (type);}
  Set the type of random address.
- #define DM_RAND_ADDR_SA(addr, type)
  Check for Static Address.
- #define DM_RAND_ADDR_RPA(addr, type)
  Check for Resolvable Private Address.

GAP Privacy Mode

Privacy Mode of this device in regards to a peer device.

- #define DM_PRIV_MODE_NETWORK 0x00
  Network privacy mode (default).
- #define DM_PRIV_MODE_DEVICE 0x01
  Device privacy mode.

DM Internal State

Connection busy or idle state

- #define DM_CONN_IDLE 0
  Connection is idle.
- #define DM_CONN_BUSY 1
  Connection is busy.

DM Internal State Flags

Connection busy/idle state bitmask.

- #define DM_IDLE_SMP_PAIR 0x0001
  SMP pairing in progress.
- #define DM_IDLE_DM_ENC 0x0002
  DM Encryption setup in progress.
- #define DM_IDLE_ATTS_DISC 0x0004
  ATTS service discovery in progress.
- #define DM_IDLE_APP_DISC 0x0008
  App framework service discovery in progress.
- #define DM_IDLE_USER_1 0x0010
  For use by user application.
- #define DM_IDLE_USER_2 0x0020
  For use by user application.
- #define DM_IDLE_USER_3 0x0040
  For use by user application.
- #define DM_IDLE_USER_4 0x0080
  For use by user application.

GAP Filter Policy Modes

Filter policy modes.

- #define DM_FILT_POLICY_MODE_ADV 0
  Advertising filter policy mode.
- #define DM_FILT_POLICY_MODE_SCAN 1
  Scanning filter policy mode.
- #define DM_FILT_POLICY_MODE_INIT 2
  Initiator filter policy mode.
- #define DM_FILT_POLICY_MODE_SYNC 3
  Synchronization filter policy mode.

DM Proprietary Error Codes

Internal error codes not sent in any PDU.

- #define DM_ERR_SMP_RX_PDU_LEN_EXCEEDED 0x01

  LESC key length exceeded maximum RX PDU length.
- #define DM_ERR_ATT_RX_PDU_LEN_EXCEEDED 0x02

  Configured ATT MTU exceeded maximum RX PDU length.
- #define DM_ERR_L2C_RX_PDU_LEN_EXCEEDED 0x03

  Registered COC MPS exceeded maximum RX PDU length.

DM Conn CTE states

Internal states of the DM conn CTE.

- enum {
  DM_CONN_CTE_STATE_IDLE,
  DM_CONN_CTE_STATE_INITIATING,
  DM_CONN_CTE_STATE_RESPONDING,
  DM_CONN_CTE_STATE_SAMPLING,
  DM_CONN_CTE_STATE_STARTING,
  DM_CONN_CTE_STATE_STOPPING }

DM Legacy Advertising Handle

Default handle for legacy advertising when using legacy HCI interface. In this case only one advertising set is allowed so all activity uses the same handle.

- #define DM_ADV_HANDLE_DEFAULT 0

  Default Advertising handle for legacy advertising.

DM ISO data path directions

Number of ISO data path directions

- #define **DM_ISO_NUM_DIR** 2

DM Callback Events

Events handled by the DM state machine.

- enum {
  DM_RESET_CMPL_IND = DM_CBACK_START,
  DM_ADV_START_IND,
  DM_ADV_STOP_IND,
  DM_ADV_NEW_ADDR_IND,
  DM_SCAN_START_IND,
  DM_SCAN_STOP_IND,
  DM_SCAN_REPORT_IND,
  DM_CONN_OPEN_IND,
  DM_CONN_CLOSE_IND,
  DM_CONN_UPDATE_IND,
  DM_SEC_PAIR_CMPL_IND,
  DM_SEC_PAIR_FAIL_IND,
  DM_SEC_ENCRYPT_IND,
  DM_SEC_ENCRYPT_FAIL_IND,
  DM_SEC_AUTH_REQ_IND,
  DM_SEC_KEY_IND,
  DM_SEC_LTK_REQ_IND,
  DM_SEC_PAIR_IND,
  DM_SEC_SLAVE_REQ_IND,
  DM_SEC_CALC_OOB_IND,
  DM_SEC_ECC_KEY_IND,
  DM_SEC_COMPARE_IND,
  DM_SEC_KEYPRESS_IND,
  DM_PRIV_RESOLVED_ADDR_IND,
  DM_PRIV_GENERATE_ADDR_IND,
  DM_CONN_READ_RSSI_IND,
  DM_PRIV_ADD_DEV_TO_RES_LIST_IND,
  DM_PRIV_REM_DEV_FROM_RES_LIST_IND,
  DM_PRIV_CLEAR_RES_LIST_IND,
  DM_PRIV_READ_PEER_RES_ADDR_IND,
  DM_PRIV_READ_LOCAL_RES_ADDR_IND,
  DM_PRIV_SET_ADDR_RES_ENABLE_IND,
  DM_REM_CONN_PARAM_REQ_IND,
  DM_CONN_DATA_LEN_CHANGE_IND,
  DM_CONN_WRITE_AUTH_TO_IND,
  DM_CONN_AUTH_TO_EXPIRED_IND,
  DM_PHY_READ_IND,
  DM_PHY_SET_DEF_IND,
  DM_PHY_UPDATE_IND,
  DM_ADV_SET_START_IND,
  DM_ADV_SET_STOP_IND,
  DM_SCAN_REQ_RCVD_IND,
  DM_EXT_SCAN_START_IND,
  DM_EXT_SCAN_STOP_IND,
  DM_EXT_SCAN_REPORT_IND,
  DM_PER_ADV_SET_START_IND,
  DM_PER_ADV_SET_STOP_IND,
  DM_PER_ADV_SYNC_EST_IND,
  DM_PER_ADV_SYNC_EST_FAIL_IND,
  DM_PER_ADV_SYNC_LOST_IND,
  DM_PER_ADV_SYNC_TRSF_EST_IND,
  DM_PER_ADV_SYNC_TRSF_EST_FAIL_IND,

DM_PER_ADV_SYNC_TRSF_IND,
DM_PER_ADV_SET_INFO_TRSF_IND,
DM_PER_ADV_REPORT_IND,
DM_REMOTE_FEATURES_IND,
DM_READ_REMOTE_VER_INFO_IND,
DM_CONN_IQ_REPORT_IND,
DM_CTE_REQ_FAIL_IND,
DM_CONN_CTE_RX_SAMPLE_START_IND,
DM_CONN_CTE_RX_SAMPLE_STOP_IND,
DM_CONN_CTE_TX_CFG_IND,
DM_CONN_CTE_REQ_START_IND,
DM_CONN_CTE_REQ_STOP_IND,
DM_CONN_CTE_RSP_START_IND,
DM_CONN_CTE_RSP_STOP_IND,
DM_READ_ANTENNA_INFO_IND,
DM_CIS_CIG_CONFIG_IND,
DM_CIS_CIG_REMOVE_IND,
DM_CIS_REQ_IND,
DM_CIS_OPEN_IND,
DM_CIS_CLOSE_IND,
DM_REQ_PEER_SCA_IND,
DM_ISO_DATA_PATH_SETUP_IND,
DM_ISO_DATA_PATH_REMOVE_IND,
DM_DATA_PATH_CONFIG_IND,
DM_READ_LOCAL_SUP_CODECS_IND,
DM_READ_LOCAL_SUP_CODEC_CAP_IND,
DM_READ_LOCAL_SUP_CTR_DLY_IND,
DM_BIG_START_IND,
DM_BIG_STOP_IND,
DM_BIG_SYNC_EST_IND,
DM_BIG_SYNC_EST_FAIL_IND,
DM_BIG_SYNC_LOST_IND,
DM_BIG_SYNC_STOP_IND,
DM_BIG_INFO_ADV_REPORT_IND,
DM_L2C_CMD_REJ_IND,
DM_ERROR_IND,
DM_HW_ERROR_IND,
DM_VENDOR_SPEC_IND }

DM callback events.

- #define DM_CBACK_START 0x20

  DM callback event starting value.

- #define DM_CBACK_END DM_VENDOR_SPEC_IND

  DM callback event ending value.

DM App Callback Registration

- void DmRegister (dmCback_t cback)

  Register a callback with DM for scan and advertising events.

DM Advertising Functions

Functions used to control Legacy and Extended Advertising.

- uint8_t * DmFindAdType (uint8_t adType, uint16_t dataLen, uint8_t *pData)

Find an advertising data element in the given advertising or scan response data.

- void DmAdvInit (void)

  Initialize DM legacy advertising.
- void DmExtAdvInit (void)

  Initialize DM extended advertising.
- bool_t DmAdvModeLeg (void)

  Whether DM advertising is in legacy mode.
- bool_t DmAdvModeExt (void)

  Whether DM advertising is in extended mode.
- void DmAdvConfig (uint8_t advHandle, uint8_t advType, uint8_t peerAddrType, uint8_t *pPeerAddr)

  Set the advertising parameters using the given advertising type, and peer address.
- void DmAdvSetData (uint8_t advHandle, uint8_t op, uint8_t location, uint8_t len, uint8_t *pData)

  Set the advertising or scan response data to the given data.
- void DmAdvStart (uint8_t numSets, uint8_t *pAdvHandles, uint16_t *pDuration, uint8_t *pMaxEaEvents)

  Start advertising using the given advertising set and duration.
- void DmAdvStop (uint8_t numSets, uint8_t *pAdvHandles)

  Stop advertising for the given advertising set. If the number of sets is set to 0 then all advertising sets are disabled.
- void DmAdvRemoveAdvSet (uint8_t advHandle)

  Remove an advertising set.
- void DmAdvClearAdvSets (void)

  Clear advertising sets.
- void DmAdvSetRandAddr (uint8_t advHandle, const uint8_t *pAddr)

  Set the random device address for a given advertising set.
- void DmAdvSetInterval (uint8_t advHandle, uint16_t intervalMin, uint16_t intervalMax)

  Set the minimum and maximum advertising intervals.
- void DmAdvSetChannelMap (uint8_t advHandle, uint8_t channelMap)

  Include or exclude certain channels from the advertising channel map.
- void DmAdvSetAddrType (uint8_t addrType)

  Set the local address type used while advertising. This function can be used to configure advertising to use a random address.
- bool_t DmAdvSetAdValue (uint8_t adType, uint8_t len, uint8_t *pValue, uint16_t *pAdvDataLen, uint8_t *p↩ AdvData, uint16_t advDataBufLen)

  Set the value of an advertising data element in the given advertising or scan response data. If the element already exists in the data then it is replaced with the new value. If the element does not exist in the data it is appended to it, space permitting.
- bool_t DmAdvSetName (uint8_t len, uint8_t *pValue, uint16_t *pAdvDataLen, uint8_t *pAdvData, uint16_t advDataBufLen)

  Set the device name in the given advertising or scan response data. If the name can only fit in the data if it is shortened, the name is shortened and the AD type is changed to DM_ADV_TYPE_SHORT_NAME.
- void DmDevPrivInit (void)

  Initialize device privacy module.
- void DmDevPrivStart (uint16_t changeInterval)

  Start using a private resolvable address.
- void DmDevPrivStop (void)

  Stop using a private resolvable address.
- void DmAdvUseLegacyPdu (uint8_t advHandle, bool_t useLegacyPdu)

  Set whether or not to use legacy advertising PDUs with extended advertising.
- void DmAdvOmitAdvAddr (uint8_t advHandle, bool_t omitAdvAddr)

  Set whether or not to omit advertiser's address from all PDUs (anonymous advertising).
- void DmAdvIncTxPwr (uint8_t advHandle, bool_t incTxPwr, int8_t advTxPwr)

  Set whether or not to include TxPower in extended header of advertising PDU.

- void DmAdvSetPhyParam (uint8_t advHandle, uint8_t priAdvPhy, uint8_t secAdvMaxSkip, uint8_t secAdv↩ Phy)

  Set extended advertising PHY parameters.
- void DmAdvScanReqNotifEnable (uint8_t advHandle, bool_t scanReqNotifEna)

  Set scan request notification enable.
- void DmAdvSetFragPref (uint8_t advHandle, uint8_t fragPref)

  Set fragment preference for advertising data.
- void DmAdvSetSid (uint8_t advHandle, uint8_t advSid)

  Set advertising SID for the given advertising handle.
- void DmPerAdvConfig (uint8_t advHandle)

  Set the advertising parameters for periodic advertising.
- void DmPerAdvSetData (uint8_t advHandle, uint8_t op, uint8_t len, uint8_t *pData)

  Set the advertising data to the given data for periodic advertising.
- void DmPerAdvStart (uint8_t advHandle)

  Start periodic advertising for the advertising set specified by the advertising handle.
- void DmPerAdvStop (uint8_t advHandle)

  Stop periodic advertising for the advertising set specified by the advertising handle.
- void DmPerAdvSetInterval (uint8_t advHandle, uint16_t intervalMin, uint16_t intervalMax)

  Set the minimum and maximum advertising intervals for periodic advertising.
- void DmPerAdvIncTxPwr (uint8_t advHandle, bool_t incTxPwr)

  Set whether or not to include TxPower in extended header of advertising PDU for periodic advertising.
- bool_t DmPerAdvEnabled (uint8_t advHandle)

  Get status of periodic advertising handle.
- uint16_t DmExtMaxAdvDataLen (uint8_t advType, bool_t useLegacyPdu)

  Get the maximum advertising data length supported by Controller for a given advertising type.

DM Privacy Functions

Functions for controlling Privacy.

- void DmPrivInit (void)

  Initialize DM privacy module.
- void DmPrivResolveAddr (uint8_t *pAddr, uint8_t *pIrk, uint16_t param)

  Resolve a private resolvable address. When complete the client's callback function is called with a DM_PRIV_RE↩ SOLVED_ADDR_IND event. The client must wait to receive this event before executing this function again.
- void DmPrivAddDevToResList (uint8_t addrType, const uint8_t *pIdentityAddr, uint8_t *pPeerIrk, uint8_t *p↩ LocalIrk, bool_t enableLlPriv, uint16_t param)

  Add device to resolving list. When complete the client's callback function is called with a DM_PRIV_ADD_DEV_T↩ O_RES_LIST_IND event. The client must wait to receive this event before executing this function again.
- void DmPrivRemDevFromResList (uint8_t addrType, const uint8_t *pIdentityAddr, uint16_t param)

  Remove device from resolving list. When complete the client's callback function is called with a DM_PRIV_REM_↩ DEV_FROM_RES_LIST_IND event. The client must wait to receive this event before executing this function again.
- void DmPrivClearResList (void)

  Clear resolving list. When complete the client's callback function is called with a DM_PRIV_CLEAR_RES_LIST_IND event. The client must wait to receive this event before executing this function again.
- void DmPrivReadPeerResolvableAddr (uint8_t addrType, const uint8_t *pIdentityAddr)

  HCI read peer resolvable address command. When complete the client's callback function is called with a DM_P↩ RIV_READ_PEER_RES_ADDR_IND event. The client must wait to receive this event before executing this function again.
- void DmPrivReadLocalResolvableAddr (uint8_t addrType, const uint8_t *pIdentityAddr)

Read local resolvable address command. When complete the client's callback function is called with a DM_PRIV_READ_LOCAL_RES_ADDR_IND event. The client must wait to receive this event before executing this function again.

- void DmPrivSetAddrResEnable (bool_t enable)

  Enable or disable address resolution in LL. When complete the client's callback function is called with a DM_PRIV_SET_ADDR_RES_ENABLE_IND event. The client must wait to receive this event before executing this function again.
- void DmPrivSetResolvablePrivateAddrTimeout (uint16_t rpaTimeout)

  Set resolvable private address timeout command.
- void DmPrivSetPrivacyMode (uint8_t addrType, const uint8_t *pIdentityAddr, uint8_t mode)

  Set privacy mode for a given entry in the resolving list.
- void DmPrivGenerateAddr (uint8_t *pIrk, uint16_t param)

  Generate a Resolvable Private Address (RPA).
- bool_t DmLlPrivEnabled (void)

  Whether LL Privacy is enabled.

DM Scanner Functions

Functions for controlling Legacy and Extended Scanner behavior.

- void DmScanInit (void)

  Initialize DM legacy scanning.
- void DmExtScanInit (void)

  Initialize DM extended scanning.
- void DmPastInit (void)

  Initialize DM Periodic Advertising Sync Transfer (PAST) module.
- void DmConnCteInit (void)

  Initialize DM Connection Constant Tone Extension (CTE) module.
- bool_t DmScanModeLeg (void)

  Whether DM scanning is in legacy mode.
- bool_t DmScanModeExt (void)

  Whether DM scanning is in extended mode.
- void DmScanStart (uint8_t scanPhys, uint8_t mode, const uint8_t *pScanType, bool_t filterDup, uint16_t duration, uint16_t period)

  Start scanning on the given PHYs.
- void DmScanStop (void)

  Stop scanning.
- void DmScanSetInterval (uint8_t scanPhys, uint16_t *pScanInterval, uint16_t *pScanWindow)

  Set the scan interval and window for the specified PHYs.
- void DmScanSetAddrType (uint8_t addrType)

  Set the local address type used while scanning. This function can be used to configure scanning to use a random address.
- dmSyncId_t DmSyncStart (uint8_t advSid, uint8_t advAddrType, const uint8_t *pAdvAddr, uint16_t skip, uint16_t syncTimeout)

  Synchronize with periodic advertising from the given advertiser, and start receiving periodic advertising packets.
- void DmSyncStop (dmSyncId_t syncId)

  Stop reception of the periodic advertising identified by the given sync identifier.
- void DmSyncSetEncrypt (uint16_t syncHandle, bool_t encrypt)

  Set the encryption mode of the Broadcast Isochronous Group (BIG) corresponding to the periodic advertising train identified by the sync handle.
- bool_t DmSyncEncrypted (uint16_t syncHandle)

Get the encryption mode of the Broadcast Isochronous Group (BIG) corresponding to the periodic advertising train identified by the sync handle.

- bool_t DmSyncEnabled (uint16_t syncHandle)

  Get status of sync identified by the handle.
- void DmSyncInitialRptEnable (bool_t enable)

  DM enable or disable initial periodic advertisement reporting.
- void DmBigSyncStart (uint8_t bigHandle, uint16_t syncHandle, uint8_t mse, uint16_t bigSyncTimeout, uint8_t numBis, uint8_t *pBis)

  Synchronize to a Broadcast Isochronous Group (BIG) described in the periodic advertising train specified by the sync handle.
- void DmBigSyncStop (uint8_t bigHandle)

  Stop synchronizing or cancel the process of synchronizing to the Broadcast Isochronous Group (BIG) identified by the handle.
- bool_t DmBisSyncInUse (uint16_t handle)

  For internal use only. Return TRUE if the BIS sync is in use.
- void DmBigSyncSetBcastCode (uint8_t bigHandle, bool_t encrypt, bool_t authen, uint8_t *pBcastCode)

  Set the Broadcast Code for the given Broadcast Isochronous Group (BIG).
- void DmBigSyncSetSecLevel (uint8_t bigHandle, uint8_t secLevel)

  Set the security level of the LE Security Mode 3 for the given Broadcast Isochronous Group (BIG).
- uint8_t DmBigSyncGetSecLevel (uint16_t handle)

  Get the security level of the LE Security Mode 3 for the given Broadcast Isochronous Group (BIG) connection handle.
- void DmBisMasterInit (void)

  Initialize DM BIS manager for operation as master.
- void DmAddDeviceToPerAdvList (uint8_t advAddrType, uint8_t *pAdvAddr, uint8_t advSid)

  Add device to periodic advertiser list.
- void DmRemoveDeviceFromPerAdvList (uint8_t advAddrType, uint8_t *pAdvAddr, uint8_t advSid)

  DM remove device from periodic advertiser list.
- void DmClearPerAdvList (void)

  DM clear periodic advertiser list.
- void DmPastRptRcvEnable (dmSyncId_t syncId, bool_t enable)

  Enable or disable reports for the periodic advertising identified by the sync id.
- void DmPastSyncTrsf (dmConnId_t connId, uint16_t serviceData, dmSyncId_t syncId)

  Send synchronization information about the periodic advertising identified by the sync id to a connected device.
- void DmPastSetInfoTrsf (dmConnId_t connId, uint16_t serviceData, uint8_t advHandle)

  Send synchronization information about the periodic advertising in an advertising set to a connected device.
- void DmPastConfig (dmConnId_t connId, uint8_t mode, uint16_t skip, uint16_t syncTimeout, uint8_t cteType)

  Specify how the Controller should process periodic advertising synchronization information received from the device identified by the connnection handle.
- void DmPastDefaultConfig (uint8_t mode, uint16_t skip, uint16_t syncTimeout, uint8_t cteType)

  Specify the initial value for the mode, skip, timeout, and Constant Tone Extension type to be used for all subsequent connections over the LE transport.
- void DmConnCteRxSampleStart (dmConnId_t connId, uint8_t slotDurations, uint8_t switchPatternLen, uint8_t *pAntennaIDs)

  Enable sampling received CTE fields on the specified connection, and configure the antenna switching pattern, and switching and sampling slot durations to be used.
- void DmConnCteRxSampleStop (dmConnId_t connId)

  Disable sampling received CTE fields on the specified connection.
- void DmConnCteTxConfig (dmConnId_t connId, uint8_t cteTypeBits, uint8_t switchPatternLen, uint8_t *p↩ AntennaIDs)

  Configure the antenna switching pattern, and permitted CTE types used for transmitting CTEs requested by the peer device on the specified connection.
- void DmConnCteReqStart (dmConnId_t connId, uint16_t cteReqInt, uint8_t reqCteLen, uint8_t reqCteType)

Initiate the CTE Request procedure on the specified connection.

- void DmConnCteReqStop (dmConnId_t connId)

  Stop initiating the CTE Request procedure on the specified connection.
- void DmConnCteRspStart (dmConnId_t connId)

  Start responding to LL_CTE_REQ PDUs with LL_CTE_RSP PDUs on the specified connection.
- void DmConnCteRspStop (dmConnId_t connId)

  Stop responding to LL_CTE_REQ PDUs with LL_CTE_RSP PDUs on the specified connection.
- uint8_t DmConnCteGetReqState (dmConnId_t connId)

  Returns the device manager's CTE request state for a given connection.
- uint8_t DmConnCteGetRspState (dmConnId_t connId)

  Returns the device manager's CTE response state for a given connection.
- void DmReadAntennaInfo (void)

  Read the switching rates, the sampling rates, the number of antennae, and the maximum length of a transmitted Constant Tone Extension supported by the Controller.

DM Connection Functions

Functions for forming connections and managing connection behavior and parameter updates.

- void DmConnInit (void)

  Initialize DM connection manager.
- void DmConnMasterInit (void)

  Initialize DM connection manager for operation as legacy master.
- void DmExtConnMasterInit (void)

  Initialize DM connection manager for operation as extended master.
- void DmConnSlaveInit (void)

  Initialize DM connection manager for operation as legacy slave.
- void DmExtConnSlaveInit (void)

  Initialize DM connection manager for operation as extended slave.
- void DmConnRegister (uint8_t clientId, dmCback_t cback)

  Register with the DM connection manager.
- dmConnId_t DmConnOpen (uint8_t clientId, uint8_t initPhys, uint8_t addrType, uint8_t *pAddr)

  Open a connection to a peer device with the given address.
- void DmConnCancelOpen (void)

  Abort connection open operation.
- void DmConnClose (uint8_t clientId, dmConnId_t connId, uint8_t reason)

  Close the connection with the give connection identifier.
- dmConnId_t DmConnAccept (uint8_t clientId, uint8_t advHandle, uint8_t advType, uint16_t duration, uint8_t maxEaEvents, uint8_t addrType, uint8_t *pAddr)

  Accept a connection from the given peer device by initiating directed advertising.
- void DmConnUpdate (dmConnId_t connId, hciConnSpec_t *pConnSpec)

  Update the connection parameters of an open connection.
- void DmConnSetScanInterval (uint16_t scanInterval, uint16_t scanWindow)

  Set the scan interval and window for connections to be created with DmConnOpen().
- void DmExtConnSetScanInterval (uint8_t initPhys, uint16_t *pScanInterval, uint16_t *pScanWindow)

  Set the scan interval and window for extended connections to be created with DmConnOpen().
- void DmConnSetConnSpec (hciConnSpec_t *pConnSpec)

  Set the connection spec parameters for connections to be created with DmConnOpen().
- void DmExtConnSetConnSpec (uint8_t initPhys, hciConnSpec_t *pConnSpec)

  Set the extended connection spec parameters for extended connections to be created with DmConnOpen().

- void DmConnSetAddrType (uint8_t addrType)

  Set the local address type used for connections created with DmConnOpen().
- void DmConnSetIdle (dmConnId_t connId, uint16_t idleMask, uint8_t idle)

  Configure a bit in the connection idle state mask as busy or idle.
- uint16_t DmConnCheckIdle (dmConnId_t connId)

  Check if a connection is idle.
- void DmConnReadRssi (dmConnId_t connId)

  Read RSSI of a given connection.
- void DmRemoteConnParamReqReply (dmConnId_t connId, hciConnSpec_t *pConnSpec)

  Reply to the HCI remote connection parameter request event. This command is used to indicate that the Host has accepted the remote device's request to change connection parameters.
- void DmRemoteConnParamReqNegReply (dmConnId_t connId, uint8_t reason)

  Negative reply to the HCI remote connection parameter request event. This command is used to indicate that the Host has rejected the remote device's request to change connection parameters.
- void DmConnSetDataLen (dmConnId_t connId, uint16_t txOctets, uint16_t txTime)

  Set data length for a given connection.
- uint8_t DmConnRole (dmConnId_t connId)

  Return the connection role indicating master or slave.
- void DmWriteAuthPayloadTimeout (dmConnId_t connId, uint16_t timeout)

  Set authenticated payload timeout for a given connection.
- void DmConnRequestPeerSca (dmConnId_t connId)

  Request the Sleep Clock Accuracy (SCA) of a peer device.

DM CIS Functions

Functions for forming and managing Connected Isochronous Stream (CIS) streams.

- void DmCisInit (void)

  Initialize DM Connected Isochronous Stream (CIS) manager.
- void DmCisMasterInit (void)

  Initialize DM Connected Isochronous Stream (CIS) manager for operation as master.
- void DmCisSlaveInit (void)

  Initialize DM Connected Isochronous Stream (CIS) manager for operation as slave.
- void DmCisCigSetSduInterval (uint8_t cigId, uint32_t sduIntervalMToS, uint32_t sduIntervalSToM)

  Set the interval, in microseconds, of periodic SDUs for the given Connected Isochronous Group (CIG).
- void DmCisCigSetSca (uint8_t cigId, uint8_t sca)

  Set the slaves clock accuracy for the given Connected Isochronous Group (CIG).
- void DmCisCigSetPackingFraming (uint8_t cigId, uint8_t packing, uint8_t framing)

  Set the packing scheme and framing format for the given Connected Isochronous Group (CIG).
- void DmCisCigSetTransLatInterval (uint8_t cigId, uint16_t transLatMToS, uint16_t transLatSToM)

  Set the maximum transport latency, in microseconds, for the given Connected Isochronous Group (CIG).
- void DmCisCigConfig (uint8_t cigId, dmConnId_t numCis, HciCisCisParams_t *pCisParam)

  Set the parameters of one or more Connected Isochronous Streams (CISes) that are associated with the given Connected Isochronous Group (CIG).
- void DmCisCigRemove (uint8_t cigId)

  Remove all the Connected Isochronous Streams (CISes) associated with the given Connected Isochronous Group (CIG).
- void DmCisOpen (uint8_t numCis, uint16_t *pCisHandle, dmConnId_t *pConnId)

  Create one or more Connected Isochronous Streams (CISes) using the connections identified by the ACL connection handles.

- void DmCisAccept (uint16_t handle)

  Inform the Controller to accept the request for the Connected Isochronous Stream (CIS) that is identified by the connection handle.
- void DmCisReject (uint16_t handle, uint8_t reason)

  Inform the Controller to reject the request for the Connected Isochronous Stream (CIS) that is identified by the connection handle.
- void DmCisClose (uint16_t handle, uint8_t reason)

  Close the Connected Isochronous Stream (CIS) connection with the given handle.
- uint8_t DmCisIdByHandle (uint16_t handle)

  For internal use only. Find the Connected Isochronous Stream (CIS) ID with matching handle.
- uint16_t DmCisHandleById (uint8_t cigId, uint8_t cisId)

  For internal use only. Find the Connected Isochronous Stream (CIS) handle with matching CIG and CIS identifiers.
- bool_t DmCisConnInUse (uint16_t handle)

  For internal use only. Return TRUE if the Connected Isochronous Stream (CIS) connection is in use.
- uint8_t DmCisConnRole (uint16_t handle)

  For internal use only. Return the CIS connection role indicating master or slave.
- bool_t DmCisCigInUse (uint8_t cigId)

  For internal use only. Return TRUE if Connected Isochronous Group (CIG) is in use.
- bool_t DmCisInUse (uint8_t cigId, uint8_t cisId)

  For internal use only. Return TRUE if the Connected Isochronous Stream (CIS) connection is in use.

DM BIS Functions

Functions for forming and managing Broadcast Isochronous Stream (BIS) streams and synchronization.

- void DmBisSlaveInit (void)

  Initialize DM BIS manager for operation as slave.
- void DmBigStart (uint8_t bigHandle, uint8_t advHandle, uint8_t numBis, uint32_t sduInterUsec, uint16_t maxSdu, uint16_t mtlMs, uint8_t rtn)

  Start a Broadcast Isochronous Group (BIG) with one or more Broadcast Isochronous Streams (BISes).
- void DmBigStop (uint8_t bigHandle, uint8_t reason)

  Stop a Broadcast Isochronous Group (BIG) identified for the given handle.
- bool_t DmBisInUse (uint16_t handle)

  For internal use only. Return TRUE if the BIS is in use.
- void DmBigSetPhy (uint8_t bigHandle, uint8_t phyBits)

  Set the PHYs used for transmission of PDUs of Broadcast Isochronous Streams (BISes) in Broadcast Isochronous Group (BIG).
- void DmBigSetPackingFraming (uint8_t bigHandle, uint8_t packing, uint32_t framing)

  Set the packing scheme and framing format for the given Broadcast Isochronous Group (BIG).
- void DmBigSetBcastCode (uint8_t bigHandle, bool_t encrypt, bool_t authen, uint8_t *pBcastCode)

  Set the Broadcast Code for the given Broadcast Isochronous Group (BIG).
- void DmBigSetSecLevel (uint8_t bigHandle, uint8_t secLevel)

  Set the security level of the LE Security Mode 3 for the given Broadcast Isochronous Group (BIG).
- uint8_t DmBigGetSecLevel (uint16_t handle)

  Get the security level of the LE Security Mode 3 for the given Broadcast Isochronous Group (BIG) connection handle.

DM Isochronous (ISO) Functions

Functions for setting up and managing isochronous data path between the Host and the Controller.

- void DmIsoInit (void)

  Initialize DM ISO manager.
- void DmIsoRegister (hciIsoCback_t cisCback, hciIsoCback_t bisCback)

  Register CIS and BIS callbacks for the HCI ISO data path.
- void DmIsoDataPathSetup (HciIsoSetupDataPath_t *pDataPathParam)

  Setup the isochronous data path between the Host and the Controller for an established Connected Isochronous Stream (CIS) or Broadcast Isochronous Stream (BIS) identified by the connection handle parameter.
- void DmIsoDataPathRemove (uint16_t handle, uint8_t directionBits)

  Remove the input and/or output data path(s) associated with a Connected Isochronous Stream (CIS) or Broadcast Isochronous Stream (BIS) identified by the connection handle parameter.
- void DmDataPathConfig (HciConfigDataPath_t *pDataPathParam)

  Request the Controller to configure the data transport path in a given direction between the Controller and the Host.
- void DmReadLocalSupCodecs (void)

  Read a list of the codecs supported by the Controller, as well as vendor specific codecs, which are defined by an individual manufacturer.
- void DmReadLocalSupCodecCap (HciReadLocalSupCodecCaps_t *pCodecParam)

  Read a list of codec capabilities supported by the Controller for a given codec.
- void DmReadLocalSupCtrDly (HciReadLocalSupControllerDly_t *pDelayParam)

  Read the range of supported Controller delays for the codec specified by Codec ID on a given transport type specified by Logical Transport Type, in the direction specified by Direction, and with the codec configuration specified by Codec Configuration.
- void DmSendIsoData (uint16_t handle, uint16_t len, uint8_t *pData)

  Send ISO Data packet.

DM PHY Control Functions

Functions for setting PHY preferences.

- void DmSetDefaultPhy (uint8_t allPhys, uint8_t txPhys, uint8_t rxPhys)

  Set the preferred values for the transmitter PHY and receiver PHY for all subsequent connections.
- void DmReadPhy (dmConnId_t connId)

  Read the current transmitter PHY and receiver PHY for a given connection.
- void DmSetPhy (dmConnId_t connId, uint8_t allPhys, uint8_t txPhys, uint8_t rxPhys, uint16_t phyOptions)

  Set the PHY preferences for a given connection.
- void DmPhyInit (void)

  Initialize DM PHY.

DM Device Functions

Device control functions

- void DmDevReset (void)

  Reset the device.
- void DmDevSetRandAddr (uint8_t *pAddr)

  Set the random address to be used by the local device.
- void DmDevWhiteListAdd (uint8_t addrType, uint8_t *pAddr)

  Add a peer device to the white list. Note that this function cannot be called while advertising, scanning, or connecting with white list filtering active.
- void DmDevWhiteListRemove (uint8_t addrType, uint8_t *pAddr)

  Remove a peer device from the white list. Note that this function cannot be called while advertising, scanning, or connecting with white list filtering active.
- void DmDevWhiteListClear (void)

  Clear the white list. Note that this function cannot be called while advertising, scanning, or connecting with white list filtering active.
- bool_t DmDevSetFilterPolicy (uint8_t mode, uint8_t policy)

  Set the Advertising, Scanning or Initiator filter policy.
- bool_t DmDevSetExtFilterPolicy (uint8_t advHandle, uint8_t mode, uint8_t policy)

  Set the Advertising filter policy for the given advertising, Scanning or Initiator filter policy.
- void DmDevVsInit (uint8_t param)

  Vendor-specific controller initialization function.

DM Security Functions

Functions for accessing and controlling security configuration of device.

- void DmSecInit (void)

  Initialize DM security.
- void DmSecLescInit (void)

  Initialize DM LE Secure Connections security.
- void DmSecPairReq (dmConnId_t connId, uint8_t oob, uint8_t auth, uint8_t iKeyDist, uint8_t rKeyDist)

  This function is called by a master device to initiate pairing.
- void DmSecPairRsp (dmConnId_t connId, uint8_t oob, uint8_t auth, uint8_t iKeyDist, uint8_t rKeyDist)

  This function is called by a slave device to proceed with pairing after a DM_SEC_PAIR_IND event is received.
- void DmSecCancelReq (dmConnId_t connId, uint8_t reason)

  This function is called to cancel the pairing process.
- void DmSecAuthRsp (dmConnId_t connId, uint8_t authDataLen, uint8_t *pAuthData)

  This function is called in response to a DM_SEC_AUTH_REQ_IND event to provide PIN or OOB data during pairing.
- void DmSecSlaveReq (dmConnId_t connId, uint8_t auth)

  This function is called by a slave device to request that the master initiates pairing or link encryption.
- void DmSecEncryptReq (dmConnId_t connId, uint8_t secLevel, dmSecLtk_t *pLtk)

  This function is called by a master device to initiate link encryption.
- void DmSecLtkRsp (dmConnId_t connId, bool_t keyFound, uint8_t secLevel, uint8_t *pKey)

  This function is called by a slave in response to a DM_SEC_LTK_REQ_IND event to provide the long term key used for encryption.
- void DmSecSetLocalCsrk (uint8_t *pCsrk)

  This function sets the local CSRK used by the device.
- void DmSecSetLocalIrk (uint8_t *pIrk)

This function sets the local IRK used by the device.

- void DmSecGenerateEccKeyReq (void)

  This function generates an ECC key for use with LESC security.
- void DmSecSetEccKey (secEccKey_t *pKey)

  This function sets the ECC key for use with LESC security.
- secEccKey_t * DmSecGetEccKey (void)

  This function gets the local ECC key for use with LESC security.
- void DmSecSetDebugEccKey (void)

  This function sets the ECC key for use with LESC security to standard debug keys values.
- void DmSecSetOob (dmConnId_t connId, dmSecLescOobCfg_t *pConfig)

  This function configures the DM to use OOB pairing for the given connection. The pRand and pConfirm contain the Random and Confirm values exchanged via out of band methods.
- void DmSecCalcOobReq (uint8_t *pRand, uint8_t *pPubKeyX)

  This function calculates the local random and confirm values used in LESC OOB pairing. The operation's result is posted as a DM_SEC_CALC_OOB_IND event to the application's DM callback handler. The local rand and confirm values are exchanged with the peer via out-of-band (OOB) methods and passed into the DmSecSetOob after DM↩ _CONN_OPEN_IND.
- void DmSecCompareRsp (dmConnId_t connId, bool_t valid)

  This function is called by the application in response to a DM_SEC_COMPARE_IND event. The valid parameter indicates if the compare value of the DM_SEC_COMPARE_IND was valid.
- uint32_t DmSecGetCompareValue (uint8_t *pConfirm)

  This function returns the 6-digit compare value for the specified 128-bit confirm value.

DM Internal Functions

Functions called internally by the stack.

- uint8_t DmLlAddrType (uint8_t addrType)

  Map an address type to a type used by LL.
- uint8_t DmHostAddrType (uint8_t addrType)

  Map an address type to a type used by Host.
- uint16_t DmSizeOfEvt (dmEvt_t *pDmEvt)

  Return size of a DM callback event.
- void DmL2cConnUpdateCnf (uint16_t handle, uint16_t reason)

  For internal use only. L2C calls this function to send the result of an L2CAP connection update response to DM.
- void DmL2cCmdRejInd (uint16_t handle, uint16_t result)

  For internal use only. L2C calls this function to send the result of an L2CAP Command Reject up to the application.
- void DmL2cConnUpdateInd (uint8_t identifier, uint16_t handle, hciConnSpec_t *pConnSpec)

  For internal use only. L2C calls this function when it receives a connection update request from a peer device.
- dmConnId_t DmConnIdByHandle (uint16_t handle)

  For internal use only. Find the connection ID with matching handle.
- bool_t DmConnInUse (dmConnId_t connId)

  For internal use only. Return TRUE if the connection is in use.
- uint8_t DmConnActiveCount (void)

  *Count active connections *.*
- uint8_t DmConnPeerAddrType (dmConnId_t connId)

  For internal use only. Return the peer address type.
- uint8_t * DmConnPeerAddr (dmConnId_t connId)

  For internal use only. Return the peer device address.
- uint8_t DmConnLocalAddrType (dmConnId_t connId)

For internal use only. Return the local address type.

- uint8_t * DmConnLocalAddr (dmConnId_t connId)

  For internal use only. Return the local address.
- uint8_t * DmConnPeerRpa (dmConnId_t connId)

  For internal use only. Return the peer resolvable private address (RPA).
- uint8_t * DmConnLocalRpa (dmConnId_t connId)

  For internal use only. Return the local resolvable private address (RPA).
- uint8_t DmConnSecLevel (dmConnId_t connId)

  For internal use only. Return the security level of the connection.
- void DmSmpEncryptReq (dmConnId_t connId, uint8_t secLevel, uint8_t *pKey)

  For internal use only. This function is called by SMP to request encryption.
- void DmSmpCbackExec (dmEvt_t *pDmEvt)

  For internal use only. Execute DM callback from SMP procedures.
- uint8_t * DmSecGetLocalCsrk (void)

  For internal use only. This function gets the local CSRK used by the device.
- uint8_t * DmSecGetLocalIrk (void)

  For internal use only. This function gets the local IRK used by the device.
- void DmReadRemoteFeatures (dmConnId_t connId)

  For internal use only. Read the features of the remote device.
- void DmReadRemoteVerInfo (dmConnId_t connId)

  Read the version info of the remote device.
- void DmDisableSlaveLatency (dmConnId_t connId, bool_t disabled)

  Disable Slave Latency.
- void DmOverruleRemoteMaxRxOctetsAndTime (dmConnId_t connId, uint16_t maxRxOctetsRemote, uint16_t maxRxTimeRemote)

  Over rule Remote Maximum Rx octets.
- void HciVsdSetDeviceAddress (uint8_t *pAddr)

  Set device address.
- void HciVsdSetTransmitPower (int8_t transmitPower)

  Set transmit power.
- void HciCmndVsdSetLeMetaVSDEvent (uint8_t event)

  Set event notification bit.
- void HciCmndVsdResetLeMetaVSDEvent (uint8_t event)

  Reset event notification bit.

1.2.1 Detailed Description

1.2.2 Macro Definition Documentation

1.2.2.1 DM_RAND_ADDR_SA

```
#define DM_RAND_ADDR_SA(
              addr,
              type )
```

Value:

```
(((type) == DM_ADDR_RANDOM) && \
                                        (DM_RAND_ADDR_GET((addr)) ==
     DM_RAND_ADDR_STATIC))
```

Check for Static Address.

Definition at line 420 of file dm_api.h.

1.2.2.2 DM_RAND_ADDR_RPA

```
#define DM_RAND_ADDR_RPA(
                addr,
                type )
```

Value:

```
(((type) == DM_ADDR_RANDOM) && \
                                        (DM_RAND_ADDR_GET((addr)) ==
      DM_RAND_ADDR_RESOLV))
```

Check for Resolvable Private Address.

Definition at line 424 of file dm_api.h.

1.2.3 Enumeration Type Documentation

1.2.3.1 anonymous enum

```
anonymous enum
```

Enumerator

DM_CONN_CTE_STATE_IDLE	Idle
DM_CONN_CTE_STATE_INITIATING	Initiating CTE request
DM_CONN_CTE_STATE_RESPONDING	Responding to CTE request
DM_CONN_CTE_STATE_SAMPLING	Sampling received CTE
DM_CONN_CTE_STATE_STARTING	Starting CTE request, CTE response or sampling received CTE
DM_CONN_CTE_STATE_STOPPING	Stopping CTE request, CTE response or sampling received CTE

Definition at line 481 of file dm_api.h.

```
482 {
483   DM_CONN_CTE_STATE_IDLE,                   /*!< Idle */
484   DM_CONN_CTE_STATE_INITIATING,             /*!< Initiating CTE request */
485   DM_CONN_CTE_STATE_RESPONDING,             /*!< Responding to CTE request */
486   DM_CONN_CTE_STATE_SAMPLING,               /*!< Sampling received CTE */
487   DM_CONN_CTE_STATE_STARTING,               /*!< Starting CTE request, CTE
       response or sampling received CTE */
488   DM_CONN_CTE_STATE_STOPPING,               /*!< Stopping CTE request, CTE
       response or sampling received CTE */
489 };
```

1.2.3.2 anonymous enum

```
anonymous enum
```

DM callback events.

Enumerator

DM_RESET_CMPL_IND	Reset complete.
DM_ADV_START_IND	Advertising started.
DM_ADV_STOP_IND	Advertising stopped.
DM_ADV_NEW_ADDR_IND	New resolvable address has been generated.
DM_SCAN_START_IND	Scanning started.
DM_SCAN_STOP_IND	Scanning stopped.
DM_SCAN_REPORT_IND	Scan data received from peer device.
DM_CONN_OPEN_IND	Connection opened.
DM_CONN_CLOSE_IND	Connection closed.
DM_CONN_UPDATE_IND	Connection update complete.
DM_SEC_PAIR_CMPL_IND	Pairing completed successfully.
DM_SEC_PAIR_FAIL_IND	Pairing failed or other security failure.
DM_SEC_ENCRYPT_IND	Connection encrypted.
DM_SEC_ENCRYPT_FAIL_IND	Encryption failed.
DM_SEC_AUTH_REQ_IND	PIN or OOB data requested for pairing.
DM_SEC_KEY_IND	Security key indication.
DM_SEC_LTK_REQ_IND	LTK requested for encyption.
DM_SEC_PAIR_IND	Incoming pairing request from master.
DM_SEC_SLAVE_REQ_IND	Incoming security request from slave.
DM_SEC_CALC_OOB_IND	Result of OOB Confirm Calculation Generation.
DM_SEC_ECC_KEY_IND	Result of ECC Key Generation.
DM_SEC_COMPARE_IND	Result of Just Works/Numeric Comparison Compare Value Calculation.
DM_SEC_KEYPRESS_IND	Keypress indication from peer in passkey security.
DM_PRIV_RESOLVED_ADDR_IND	Private address resolved.
DM_PRIV_GENERATE_ADDR_IND	Private resolvable address generated.
DM_CONN_READ_RSSI_IND	Connection RSSI read.
DM_PRIV_ADD_DEV_TO_RES_LIST_IND	Device added to resolving list.
DM_PRIV_REM_DEV_FROM_RES_LIST_IND	Device removed from resolving list.
DM_PRIV_CLEAR_RES_LIST_IND	Resolving list cleared.
DM_PRIV_READ_PEER_RES_ADDR_IND	Peer resolving address read.
DM_PRIV_READ_LOCAL_RES_ADDR_IND	Local resolving address read.
DM_PRIV_SET_ADDR_RES_ENABLE_IND	Address resolving enable set.
DM_REM_CONN_PARAM_REQ_IND	Remote connection parameter requested.
DM_CONN_DATA_LEN_CHANGE_IND	Data length changed.
DM_CONN_WRITE_AUTH_TO_IND	Write authenticated payload complete.
DM_CONN_AUTH_TO_EXPIRED_IND	Authenticated payload timeout expired.
DM_PHY_READ_IND	Read PHY.
DM_PHY_SET_DEF_IND	Set default PHY.
DM_PHY_UPDATE_IND	PHY update.
DM_ADV_SET_START_IND	Advertising set(s) started.
DM_ADV_SET_STOP_IND	Advertising set(s) stopped.
DM_SCAN_REQ_RCVD_IND	Scan request received.
DM_EXT_SCAN_START_IND	Extended scanning started.
DM_EXT_SCAN_STOP_IND	Extended scanning stopped.
DM_EXT_SCAN_REPORT_IND	Extended scan data received from peer device.
DM_PER_ADV_SET_START_IND	Periodic advertising set started.

Enumerator

DM_PER_ADV_SET_STOP_IND	Periodic advertising set stopped.
DM_PER_ADV_SYNC_EST_IND	Periodic advertising sync established.
DM_PER_ADV_SYNC_EST_FAIL_IND	Periodic advertising sync establishment failed.
DM_PER_ADV_SYNC_LOST_IND	Periodic advertising sync lost.
DM_PER_ADV_SYNC_TRSF_EST_IND	Periodic advertising sync transfer established.
DM_PER_ADV_SYNC_TRSF_EST_FAIL_IND	Periodic advertising sync transfer establishment failed.
DM_PER_ADV_SYNC_TRSF_IND	Periodic advertising sync info transferred.
DM_PER_ADV_SET_INFO_TRSF_IND	Periodic advertising set sync info transferred.
DM_PER_ADV_REPORT_IND	Periodic advertising data received from peer device.
DM_REMOTE_FEATURES_IND	Remote features from peer device.
DM_READ_REMOTE_VER_INFO_IND	Remote LL version information read.
DM_CONN_IQ_REPORT_IND	IQ samples from CTE of received packet from peer device.
DM_CTE_REQ_FAIL_IND	CTE request failed.
DM_CONN_CTE_RX_SAMPLE_START_IND	Sampling received CTE started.
DM_CONN_CTE_RX_SAMPLE_STOP_IND	Sampling received CTE stopped.
DM_CONN_CTE_TX_CFG_IND	Connection CTE transmit parameters configured.
DM_CONN_CTE_REQ_START_IND	Initiating connection CTE request started.
DM_CONN_CTE_REQ_STOP_IND	Initiating connection CTE request stopped.
DM_CONN_CTE_RSP_START_IND	Responding to connection CTE request started.
DM_CONN_CTE_RSP_STOP_IND	Responding to connection CTE request stopped.
DM_READ_ANTENNA_INFO_IND	Antenna information read.
DM_CIS_CIG_CONFIG_IND	CIS CIG configure complete.
DM_CIS_CIG_REMOVE_IND	CIS CIG remove complete.
DM_CIS_REQ_IND	CIS request.
DM_CIS_OPEN_IND	CIS connection opened.
DM_CIS_CLOSE_IND	CIS connection closed.
DM_REQ_PEER_SCA_IND	Request peer SCA complete.
DM_ISO_DATA_PATH_SETUP_IND	ISO data path setup complete.
DM_ISO_DATA_PATH_REMOVE_IND	ISO data path remove complete.
DM_DATA_PATH_CONFIG_IND	Data path configure complete.
DM_READ_LOCAL_SUP_CODECS_IND	Local supported codecs read.
DM_READ_LOCAL_SUP_CODEC_CAP_IND	Local supported codec capabilities read.
DM_READ_LOCAL_SUP_CTR_DLY_IND	Local supported controller delay read.
DM_BIG_START_IND	BIG started.
DM_BIG_STOP_IND	BIG stopped.
DM_BIG_SYNC_EST_IND	BIG sync established.
DM_BIG_SYNC_EST_FAIL_IND	BIG sync establishment failed.
DM_BIG_SYNC_LOST_IND	BIG sync lost.
DM_BIG_SYNC_STOP_IND	BIG sync stopped.
DM_BIG_INFO_ADV_REPORT_IND	BIG Info advertising data received from peer device.
DM_L2C_CMD_REJ_IND	L2CAP Command Reject.
DM_ERROR_IND	General error.
DM_HW_ERROR_IND	Hardware error.
DM_VENDOR_SPEC_IND	Vendor specific event.

Definition at line 515 of file dm_api.h.

```
516 {
517   DM_RESET_CMPL_IND = DM_CBACK_START,     /*!< \brief Reset complete */
518   DM_ADV_START_IND,                       /*!< \brief Advertising started */
519   DM_ADV_STOP_IND,                        /*!< \brief Advertising stopped */
520   DM_ADV_NEW_ADDR_IND,                    /*!< \brief New resolvable address has been
       generated */
521   DM_SCAN_START_IND,                      /*!< \brief Scanning started */
522   DM_SCAN_STOP_IND,                       /*!< \brief Scanning stopped */
523   DM_SCAN_REPORT_IND,                     /*!< \brief Scan data received from peer device
       */
524   DM_CONN_OPEN_IND,                       /*!< \brief Connection opened */
525   DM_CONN_CLOSE_IND,                      /*!< \brief Connection closed */
526   DM_CONN_UPDATE_IND,                     /*!< \brief Connection update complete */
527   DM_SEC_PAIR_CMPL_IND,                   /*!< \brief Pairing completed successfully */
528   DM_SEC_PAIR_FAIL_IND,                   /*!< \brief Pairing failed or other security
       failure */
529   DM_SEC_ENCRYPT_IND,                     /*!< \brief Connection encrypted */
530   DM_SEC_ENCRYPT_FAIL_IND,                /*!< \brief Encryption failed */
531   DM_SEC_AUTH_REQ_IND,                    /*!< \brief PIN or OOB data requested for
       pairing */
532   DM_SEC_KEY_IND,                         /*!< \brief Security key indication */
533   DM_SEC_LTK_REQ_IND,                     /*!< \brief LTK requested for encyption */
534   DM_SEC_PAIR_IND,                        /*!< \brief Incoming pairing request from master
      */
535   DM_SEC_SLAVE_REQ_IND,                   /*!< \brief Incoming security request from
       slave */
536   DM_SEC_CALC_OOB_IND,                    /*!< \brief Result of OOB Confirm Calculation
       Generation */
537   DM_SEC_ECC_KEY_IND,                     /*!< \brief Result of ECC Key Generation */
538   DM_SEC_COMPARE_IND,                     /*!< \brief Result of Just Works/Numeric
       Comparison Compare Value Calculation */
539   DM_SEC_KEYPRESS_IND,                    /*!< \brief Keypress indication from peer in
       passkey security */
540   DM_PRIV_RESOLVED_ADDR_IND,              /*!< \brief Private address resolved */
541   DM_PRIV_GENERATE_ADDR_IND,              /*!< \brief Private resolvable address
       generated */
542   DM_CONN_READ_RSSI_IND,                  /*!< \brief Connection RSSI read */
543   DM_PRIV_ADD_DEV_TO_RES_LIST_IND,        /*!< \brief Device added to
       resolving list */
544   DM_PRIV_REM_DEV_FROM_RES_LIST_IND,      /*!< \brief Device removed from
       resolving list */
545   DM_PRIV_CLEAR_RES_LIST_IND,             /*!< \brief Resolving list cleared */
546   DM_PRIV_READ_PEER_RES_ADDR_IND,         /*!< \brief Peer resolving address
       read */
547   DM_PRIV_READ_LOCAL_RES_ADDR_IND,        /*!< \brief Local resolving
       address read */
548   DM_PRIV_SET_ADDR_RES_ENABLE_IND,        /*!< \brief Address resolving
       enable set */
549   DM_REM_CONN_PARAM_REQ_IND,              /*!< \brief Remote connection parameter
       requested */
550   DM_CONN_DATA_LEN_CHANGE_IND,            /*!< \brief Data length changed */
551   DM_CONN_WRITE_AUTH_TO_IND,              /*!< \brief Write authenticated payload
       complete */
552   DM_CONN_AUTH_TO_EXPIRED_IND,            /*!< \brief Authenticated payload
       timeout expired */
553   DM_PHY_READ_IND,                        /*!< \brief Read PHY */
554   DM_PHY_SET_DEF_IND,                     /*!< \brief Set default PHY */
555   DM_PHY_UPDATE_IND,                      /*!< \brief PHY update */
556   DM_ADV_SET_START_IND,                   /*!< \brief Advertising set(s) started */
557   DM_ADV_SET_STOP_IND,                    /*!< \brief Advertising set(s) stopped */
558   DM_SCAN_REQ_RCVD_IND,                   /*!< \brief Scan request received */
559   DM_EXT_SCAN_START_IND,                  /*!< \brief Extended scanning started */
560   DM_EXT_SCAN_STOP_IND,                   /*!< \brief Extended scanning stopped */
561   DM_EXT_SCAN_REPORT_IND,                 /*!< \brief Extended scan data received
       from peer device */
562   DM_PER_ADV_SET_START_IND,               /*!< \brief Periodic advertising set
       started */
563   DM_PER_ADV_SET_STOP_IND,                /*!< \brief Periodic advertising set
       stopped */
564   DM_PER_ADV_SYNC_EST_IND,                /*!< \brief Periodic advertising sync
       established */
565   DM_PER_ADV_SYNC_EST_FAIL_IND,           /*!< \brief Periodic advertising sync
       establishment failed */
566   DM_PER_ADV_SYNC_LOST_IND,               /*!< \brief Periodic advertising sync
       lost */
567   DM_PER_ADV_SYNC_TRSF_EST_IND,           /*!< \brief Periodic advertising sync
       transfer established */
568   DM_PER_ADV_SYNC_TRSF_EST_FAIL_IND,      /*!< \brief Periodic advertising
       sync transfer establishment failed */
569   DM_PER_ADV_SYNC_TRSF_IND,               /*!< \brief Periodic advertising sync
       info transferred */
570   DM_PER_ADV_SET_INFO_TRSF_IND,           /*!< \brief Periodic advertising set
       sync info transferred */
571   DM_PER_ADV_REPORT_IND,                  /*!< \brief Periodic advertising data
       received from peer device */
572   DM_REMOTE_FEATURES_IND,                 /*!< \brief Remote features from peer
       device */
```

```
573   DM_READ_REMOTE_VER_INFO_IND,          /*!< \brief Remote LL version
       information read */
574   DM_CONN_IQ_REPORT_IND,                /*!< \brief IQ samples from CTE of received
       packet from peer device */
575   DM_CTE_REQ_FAIL_IND,                  /*!< \brief CTE request failed */
576   DM_CONN_CTE_RX_SAMPLE_START_IND,      /*!< \brief Sampling received CTE
       started */
577   DM_CONN_CTE_RX_SAMPLE_STOP_IND,       /*!< \brief Sampling received CTE
       stopped */
578   DM_CONN_CTE_TX_CFG_IND,               /*!< \brief Connection CTE transmit
       parameters configured */
579   DM_CONN_CTE_REQ_START_IND,            /*!< \brief Initiating connection CTE
       request started */
580   DM_CONN_CTE_REQ_STOP_IND,             /*!< \brief Initiating connection CTE
       request stopped */
581   DM_CONN_CTE_RSP_START_IND,            /*!< \brief Responding to connection CTE
       request started */
582   DM_CONN_CTE_RSP_STOP_IND,             /*!< \brief Responding to connection CTE
       request stopped */
583   DM_READ_ANTENNA_INFO_IND,             /*!< \brief Antenna information read */
584   DM_CIS_CIG_CONFIG_IND,                /*!< \brief CIS CIG configure complete */
585   DM_CIS_CIG_REMOVE_IND,                /*!< \brief CIS CIG remove complete */
586   DM_CIS_REQ_IND,                       /*!< \brief CIS request */
587   DM_CIS_OPEN_IND,                      /*!< \brief CIS connection opened */
588   DM_CIS_CLOSE_IND,                     /*!< \brief CIS connection closed */
589   DM_REQ_PEER_SCA_IND,                  /*!< \brief Request peer SCA complete */
590   DM_ISO_DATA_PATH_SETUP_IND,           /*!< \brief ISO data path setup
       complete */
591   DM_ISO_DATA_PATH_REMOVE_IND,          /*!< \brief ISO data path remove
       complete */
592   DM_DATA_PATH_CONFIG_IND,              /*!< \brief Data path configure complete
      */
593   DM_READ_LOCAL_SUP_CODECS_IND,         /*!< \brief Local supported codecs
       read */
594   DM_READ_LOCAL_SUP_CODEC_CAP_IND,      /*!< \brief Local supported codec
       capabilities read */
595   DM_READ_LOCAL_SUP_CTR_DLY_IND,        /*!< \brief Local supported
       controller delay read */
596   DM_BIG_START_IND,                     /*!< \brief BIG started */
597   DM_BIG_STOP_IND,                      /*!< \brief BIG stopped */
598   DM_BIG_SYNC_EST_IND,                  /*!< \brief BIG sync established */
599   DM_BIG_SYNC_EST_FAIL_IND,             /*!< \brief BIG sync establishment failed
       */
600   DM_BIG_SYNC_LOST_IND,                 /*!< \brief BIG sync lost */
601   DM_BIG_SYNC_STOP_IND,                 /*!< \brief BIG sync stopped */
602   DM_BIG_INFO_ADV_REPORT_IND,           /*!< \brief BIG Info advertising data
       received from peer device */
603   DM_L2C_CMD_REJ_IND,                   /*!< \brief L2CAP Command Reject */
604   DM_ERROR_IND,                         /*!< \brief General error */
605   DM_HW_ERROR_IND,                      /*!< \brief Hardware error */
606   DM_VENDOR_SPEC_IND                    /*!< \brief Vendor specific event */
607 };
```

1.2.4 Function Documentation

1.2.4.1 DmRegister()

```
void DmRegister (
              dmCback_t cback )
```

Register a callback with DM for scan and advertising events.

Parameters

cback	Client callback function.

Returns

None.

1.2.4.2 DmFindAdType()

```
uint8_t* DmFindAdType (
             uint8_t adType,
             uint16_t dataLen,
             uint8_t * pData )
```

Find an advertising data element in the given advertising or scan response data.

Parameters

adType	Advertising data element type to find.
dataLen	Data length.
pData	Pointer to advertising or scan response data.

Returns

Pointer to the advertising data element byte array or NULL if not found.

1.2.4.3 DmAdvInit()

```
void DmAdvInit (
             void  )
```

Initialize DM legacy advertising.

Returns

None.

1.2.4.4 DmExtAdvInit()

```
void DmExtAdvInit (
             void  )
```

Initialize DM extended advertising.

Returns

None.

1.2.4.5 DmAdvModeLeg()

```
bool_t DmAdvModeLeg (
            void  )
```

Whether DM advertising is in legacy mode.

Returns

TRUE if DM advertising is in legacy mode. FALSE, otherwise.

1.2.4.6 DmAdvModeExt()

```
bool_t DmAdvModeExt (
            void  )
```

Whether DM advertising is in extended mode.

Returns

TRUE if DM advertising is in extended mode. FALSE, otherwise.

1.2.4.7 DmAdvConfig()

```
void DmAdvConfig (
            uint8_t advHandle,
            uint8_t advType,
            uint8_t peerAddrType,
            uint8_t * pPeerAddr )
```

Set the advertising parameters using the given advertising type, and peer address.

Parameters

advHandle	Advertising handle.
advType	Advertising type.
peerAddrType	Peer address type.
pPeerAddr	Peer address.

Returns

None.

1.2.4.8 DmAdvSetData()

```
void DmAdvSetData (
             uint8_t advHandle,
             uint8_t op,
             uint8_t location,
             uint8_t len,
             uint8_t * pData )
```

Set the advertising or scan response data to the given data.

Parameters

advHandle	Advertising handle.
op	Data operation.
location	Data location.
len	Length of the data. Maximum length is 236 bytes.
pData	Pointer to the data.

Returns

None.

1.2.4.9 DmAdvStart()

```
void DmAdvStart (
             uint8_t numSets,
             uint8_t * pAdvHandles,
             uint16_t * pDuration,
             uint8_t * pMaxEaEvents )
```

Start advertising using the given advertising set and duration.

Parameters

numSets	Number of advertising sets to enable.
pAdvHandles	Advertising handles array.
pDuration	Advertising duration (in milliseconds) array.
pMaxEaEvents	Maximum number of extended advertising events array.

Returns

None.

1.2.4.10 DmAdvStop()

```
void DmAdvStop (
             uint8_t numSets,
             uint8_t * pAdvHandles )
```

Stop advertising for the given advertising set. If the number of sets is set to 0 then all advertising sets are disabled.

Parameters

numSets	Number of advertising sets to disable.
pAdvHandles	Advertising handles array.

Returns

None.

1.2.4.11 DmAdvRemoveAdvSet()

```
void DmAdvRemoveAdvSet (
             uint8_t advHandle )
```

Remove an advertising set.

Parameters

advHandle	Advertising handle.

Returns

None.

1.2.4.12 DmAdvClearAdvSets()

```
void DmAdvClearAdvSets (
             void  )
```

Clear advertising sets.

Returns

None.

1.2.4.13 DmAdvSetRandAddr()

```
void DmAdvSetRandAddr (
             uint8_t advHandle,
             const uint8_t * pAddr )
```

Set the random device address for a given advertising set.

Parameters

advHandle	Advertising handle.
pAddr	Random device address.

Returns

None.

1.2.4.14 DmAdvSetInterval()

```
void DmAdvSetInterval (
             uint8_t advHandle,
             uint16_t intervalMin,
             uint16_t intervalMax )
```

Set the minimum and maximum advertising intervals.

Parameters

advHandle	Advertising handle.
intervalMin	Minimum advertising interval.
intervalMax	Maximum advertising interval.

Returns

None.

1.2.4.15 DmAdvSetChannelMap()

```
void DmAdvSetChannelMap (
             uint8_t advHandle,
             uint8_t channelMap )
```

Include or exclude certain channels from the advertising channel map.

Parameters

advHandle	Advertising handle.
channelMap	Advertising channel map.

Returns

None.

1.2.4.16 DmAdvSetAddrType()

```
void DmAdvSetAddrType (
            uint8_t addrType )
```

Set the local address type used while advertising. This function can be used to configure advertising to use a random address.

Parameters

addrType	Address type.

Returns

None.

1.2.4.17 DmAdvSetAdValue()

```
bool_t DmAdvSetAdValue (
            uint8_t adType,
            uint8_t len,
            uint8_t * pValue,
            uint16_t * pAdvDataLen,
            uint8_t * pAdvData,
            uint16_t advDataBufLen )
```

Set the value of an advertising data element in the given advertising or scan response data. If the element already exists in the data then it is replaced with the new value. If the element does not exist in the data it is appended to it, space permitting.

Parameters

adType	Advertising data element type.
len	Length of the value. Maximum length is 29 bytes.
pValue	Pointer to the value.
pAdvDataLen	Advertising or scan response data length. The new length is returned in this parameter.
pAdvData	Pointer to advertising or scan response data.
advDataBufLen	Length of the advertising or scan response data buffer maintained by Application.

Returns

TRUE if the element was successfully added to the data, FALSE otherwise.

1.2.4.18 DmAdvSetName()

```
bool_t DmAdvSetName (
             uint8_t len,
             uint8_t * pValue,
             uint16_t * pAdvDataLen,
             uint8_t * pAdvData,
             uint16_t advDataBufLen )
```

Set the device name in the given advertising or scan response data. If the name can only fit in the data if it is shortened, the name is shortened and the AD type is changed to DM_ADV_TYPE_SHORT_NAME.

Parameters

len	Length of the name. Maximum length is 29 bytes.
pValue	Pointer to the name in UTF-8 format.
pAdvDataLen	Advertising or scan response data length. The new length is returned in this parameter.
pAdvData	Pointer to advertising or scan response data.
advDataBufLen	Length of the advertising or scan response data buffer maintained by Application.

Returns

TRUE if the element was successfully added to the data, FALSE otherwise.

1.2.4.19 DmDevPrivInit()

```
void DmDevPrivInit (
             void  )
```

Initialize device privacy module.

Returns

None.

1.2.4.20 DmDevPrivStart()

```
void DmDevPrivStart (
             uint16_t changeInterval )
```

Start using a private resolvable address.

Parameters

changeInterval	Interval between automatic address changes, in seconds.

Returns

None.

1.2.4.21 DmDevPrivStop()

```
void DmDevPrivStop (
             void  )
```

Stop using a private resolvable address.

Returns

None.

1.2.4.22 DmAdvUseLegacyPdu()

```
void DmAdvUseLegacyPdu (
             uint8_t advHandle,
             bool_t useLegacyPdu )
```

Set whether or not to use legacy advertising PDUs with extended advertising.

Parameters

advHandle	Advertising handle.
useLegacyPdu	Whether to use legacy advertising PDUs (default value is TRUE).

Returns

None.

1.2.4.23 DmAdvOmitAdvAddr()

```
void DmAdvOmitAdvAddr (
             uint8_t advHandle,
             bool_t omitAdvAddr )
```

Set whether or not to omit advertiser's address from all PDUs (anonymous advertising).

Parameters

advHandle	Advertising handle.
omitAdvAddr	Whether to omit advertiser's address from all PDUs (default value is FALSE).

Returns

None.

1.2.4.24 DmAdvIncTxPwr()

```
void DmAdvIncTxPwr (
             uint8_t advHandle,
             bool_t incTxPwr,
             int8_t advTxPwr )
```

Set whether or not to include TxPower in extended header of advertising PDU.

Parameters

advHandle	Advertising handle.
incTxPwr	Whether to include TxPower in extended header of advertising PDU (default value is FALSE).
advTxPwr	Advertising tx power (127 = no preference).

Returns

None.

1.2.4.25 DmAdvSetPhyParam()

```
void DmAdvSetPhyParam (
             uint8_t advHandle,
             uint8_t priAdvPhy,
             uint8_t secAdvMaxSkip,
             uint8_t secAdvPhy )
```

Set extended advertising PHY parameters.

Parameters

advHandle	Advertising handle.
priAdvPhy	Primary advertising Phy.
secAdvMaxSkip	Maximum advertising events Controller can skip before sending AUX_ADV_IND on secondary advertising channel (0 = AUX_ADV_IND will be sent prior to next advertising event).
secAdvPhy	Secondary advertising Phy.

Returns

None.

1.2.4.26 DmAdvScanReqNotifEnable()

```
void DmAdvScanReqNotifEnable (
            uint8_t advHandle,
            bool_t scanReqNotifEna )
```

Set scan request notification enable.

Parameters

advHandle	Advertising handle.
scanReqNotifEna	Scan request notification enable.

Returns

None.

1.2.4.27 DmAdvSetFragPref()

```
void DmAdvSetFragPref (
            uint8_t advHandle,
            uint8_t fragPref )
```

Set fragment preference for advertising data.

Parameters

advHandle	Advertising handle.
fragPref	Fragment preference.

Returns

None.

1.2.4.28 DmAdvSetSid()

```
void DmAdvSetSid (
            uint8_t advHandle,
            uint8_t advSid )
```

Set advertising SID for the given advertising handle.

Parameters

advHandle	Advertising handle.
advSid	Advertsing SID.

Returns

None.

1.2.4.29 DmPerAdvConfig()

```
void DmPerAdvConfig (
             uint8_t advHandle )
```

Set the advertising parameters for periodic advertising.

Parameters

advHandle	Advertising handle.

Returns

None.

1.2.4.30 DmPerAdvSetData()

```
void DmPerAdvSetData (
             uint8_t advHandle,
             uint8_t op,
             uint8_t len,
             uint8_t * pData )
```

Set the advertising data to the given data for periodic advertising.

Parameters

advHandle	Advertising handle.
op	Data operation.
len	Length of the data. Maximum length is 236 bytes.
pData	Pointer to the data.

Returns

None.

1.2.4.31 DmPerAdvStart()

```
void DmPerAdvStart (
            uint8_t advHandle )
```

Start periodic advertising for the advertising set specified by the advertising handle.

Parameters

advHandle	Advertising handle.

Returns

None.

1.2.4.32 DmPerAdvStop()

```
void DmPerAdvStop (
            uint8_t advHandle )
```

Stop periodic advertising for the advertising set specified by the advertising handle.

Parameters

advHandle	Advertising handle.

Returns

None.

1.2.4.33 DmPerAdvSetInterval()

```
void DmPerAdvSetInterval (
            uint8_t advHandle,
            uint16_t intervalMin,
            uint16_t intervalMax )
```

Set the minimum and maximum advertising intervals for periodic advertising.

Parameters

advHandle	Advertising handle.
intervalMin	Minimum advertising interval.
intervalMax	Maximum advertising interval.

Returns

None.

1.2.4.34 DmPerAdvIncTxPwr()

```
void DmPerAdvIncTxPwr (
            uint8_t advHandle,
            bool_t incTxPwr )
```

Set whether or not to include TxPower in extended header of advertising PDU for periodic advertising.

Parameters

advHandle	Advertising handle.
incTxPwr	Whether to include TxPower in extended header of advertising PDU (default value is FALSE).

Returns

None.

1.2.4.35 DmPerAdvEnabled()

```
bool_t DmPerAdvEnabled (
            uint8_t advHandle )
```

Get status of periodic advertising handle.

Parameters

advHandle	Advertising handle.

Returns

TRUE if periodic advertising is running on that handle. FALSE, otherwise.

1.2.4.36 DmExtMaxAdvDataLen()

```
uint16_t DmExtMaxAdvDataLen (
            uint8_t advType,
            bool_t useLegacyPdu )
```

Get the maximum advertising data length supported by Controller for a given advertising type.

Parameters

advType	Advertising type.
useLegacyPdu	Whether to use legacy advertising PDUs with extended advertising.

Returns

Maximum advertising data length.

1.2.4.37 DmPrivInit()

```
void DmPrivInit (
             void  )
```

Initialize DM privacy module.

Returns

None.

1.2.4.38 DmPrivResolveAddr()

```
void DmPrivResolveAddr (
             uint8_t * pAddr,
             uint8_t * pIrk,
             uint16_t param )
```

Resolve a private resolvable address. When complete the client's callback function is called with a DM_PRIV_R↩ ESOLVED_ADDR_IND event. The client must wait to receive this event before executing this function again.

Parameters

pAddr	Peer device address.
pIrk	The peer's identity resolving key.
param	Client-defined parameter returned with callback event.

Returns

None.

1.2.4.39 DmPrivAddDevToResList()

```
void DmPrivAddDevToResList (
            uint8_t addrType,
            const uint8_t * pIdentityAddr,
            uint8_t * pPeerIrk,
            uint8_t * pLocalIrk,
            bool_t enableLlPriv,
            uint16_t param )
```

Add device to resolving list. When complete the client's callback function is called with a DM_PRIV_ADD_DEV_↩ TO_RES_LIST_IND event. The client must wait to receive this event before executing this function again.

Parameters

addrType	Peer identity address type.
pIdentityAddr	Peer identity address.
pPeerIrk	The peer's identity resolving key.
pLocalIrk	The local identity resolving key.
enableLlPriv	Set to TRUE to enable address resolution in LL.
param	client-defined parameter returned with callback event.

Returns

None.

This command cannot be used when address resolution is enabled in the Controller and:

- Advertising (other than periodic advertising) is enabled,
- Scanning is enabled, or
- (Extended) Create connection or Create Sync command is outstanding.

If the local or peer IRK associated with the peer Identity Address is all zeros then the Controller will use or accept the local or peer Identity Address respectively.

Parameter 'enableLlPriv' should be set to TRUE when the last device is being added to resolving list to enable address resolution in the Controller.

1.2.4.40 DmPrivRemDevFromResList()

```
void DmPrivRemDevFromResList (
            uint8_t addrType,
            const uint8_t * pIdentityAddr,
            uint16_t param )
```

Remove device from resolving list. When complete the client's callback function is called with a DM_PRIV_R↩ EM_DEV_FROM_RES_LIST_IND event. The client must wait to receive this event before executing this function again.

Parameters

addrType	Peer identity address type.
pIdentityAddr	Peer identity address.
param	client-defined parameter returned with callback event.

Returns

None.

This command cannot be used when address resolution is enabled in the Controller and:

- Advertising (other than periodic advertising) is enabled,
- Scanning is enabled, or
- (Extended) Create connection or Create Sync command is outstanding.

1.2.4.41 DmPrivClearResList()

```
void DmPrivClearResList (
            void  )
```

Clear resolving list. When complete the client's callback function is called with a DM_PRIV_CLEAR_RES_LIST_IND event. The client must wait to receive this event before executing this function again.

Returns

None.

This command cannot be used when address resolution is enabled in the Controller and:

- Advertising (other than periodic advertising) is enabled,
- Scanning is enabled, or
- (Extended) Create connection or Create Sync command is outstanding.

Address resolution in Controller will be disabled when resolving list's cleared successfully.

1.2.4.42 DmPrivReadPeerResolvableAddr()

```
void DmPrivReadPeerResolvableAddr (
            uint8_t addrType,
            const uint8_t * pIdentityAddr )
```

HCI read peer resolvable address command. When complete the client's callback function is called with a DM_P↩RIV_READ_PEER_RES_ADDR_IND event. The client must wait to receive this event before executing this function again.

Parameters

addrType	Peer identity address type.
pIdentityAddr	Peer identity address.

Returns

None.

1.2.4.43 DmPrivReadLocalResolvableAddr()

```
void DmPrivReadLocalResolvableAddr (
            uint8_t addrType,
            const uint8_t * pIdentityAddr )
```

Read local resolvable address command. When complete the client's callback function is called with a DM_PRIV_READ_LOCAL_RES_ADDR_IND event. The client must wait to receive this event before executing this function again.

Parameters

addrType	Peer identity address type.
pIdentityAddr	Peer identity address.

Returns

None.

1.2.4.44 DmPrivSetAddrResEnable()

```
void DmPrivSetAddrResEnable (
            bool_t enable )
```

Enable or disable address resolution in LL. When complete the client's callback function is called with a DM_PRIV_SET_ADDR_RES_ENABLE_IND event. The client must wait to receive this event before executing this function again.

Parameters

enable	Set to TRUE to enable address resolution or FALSE to disable it.

Returns

None.

This command can be used at any time except when:

- Advertising (other than periodic advertising) is enabled,
- Scanning is enabled, or
- (Extended) Create connection or Create Sync command is outstanding.

1.2.4.45 DmPrivSetResolvablePrivateAddrTimeout()

```
void DmPrivSetResolvablePrivateAddrTimeout (
            uint16_t rpaTimeout )
```

Set resolvable private address timeout command.

Parameters

rpaTimeout	Timeout measured in seconds.

Returns

None.

1.2.4.46 DmPrivSetPrivacyMode()

```
void DmPrivSetPrivacyMode (
            uint8_t addrType,
            const uint8_t * pIdentityAddr,
            uint8_t mode )
```

Set privacy mode for a given entry in the resolving list.

Parameters

addrType	Peer identity address type.
pIdentityAddr	Peer identity address.
mode	Privacy mode (by default, network privacy mode is used).

Returns

None.

This command can be used at any time except when:

- Advertising (other than periodic advertising) is enabled,

- Scanning is enabled, or
- (Extended) Create connection or Create Sync command is outstanding.

1.2.4.47 DmPrivGenerateAddr()

```
void DmPrivGenerateAddr (
             uint8_t * pIrk,
             uint16_t param )
```

Generate a Resolvable Private Address (RPA).

Parameters

plrk	The identity resolving key.
param	Client-defined parameter returned with callback event.

Returns

None.

1.2.4.48 DmLlPrivEnabled()

```
bool_t DmLlPrivEnabled (
             void  )
```

Whether LL Privacy is enabled.

Returns

TRUE if LL Privacy is enabled. FALSE, otherwise.

1.2.4.49 DmScanInit()

```
void DmScanInit (
             void  )
```

Initialize DM legacy scanning.

Returns

None.

1.2.4.50 DmExtScanInit()

```
void DmExtScanInit (
             void  )
```

Initialize DM extended scanning.

Returns

None.

1.2.4.51 DmPastInit()

```
void DmPastInit (
             void  )
```

Initialize DM Periodic Advertising Sync Transfer (PAST) module.

Returns

None.

1.2.4.52 DmConnCteInit()

```
void DmConnCteInit (
             void  )
```

Initialize DM Connection Constant Tone Extension (CTE) module.

Returns

None.

1.2.4.53 DmScanModeLeg()

```
bool_t DmScanModeLeg (
             void  )
```

Whether DM scanning is in legacy mode.

Returns

TRUE if DM scanning is in legacy mode. FALSE, otherwise.

1.2.4.54 DmScanModeExt()

```
bool_t DmScanModeExt (
            void  )
```

Whether DM scanning is in extended mode.

Returns

TRUE if DM scanning is in extended mode. FALSE, otherwise.

1.2.4.55 DmScanStart()

```
void DmScanStart (
            uint8_t scanPhys,
            uint8_t mode,
            const uint8_t * pScanType,
            bool_t filterDup,
            uint16_t duration,
            uint16_t period )
```

Start scanning on the given PHYs.

Parameters

scanPhys	Scanner PHYs.
mode	Discoverability mode.
pScanType	Scan type array.
filterDup	Filter duplicates. Set to TRUE to filter duplicate responses received from the same device. Set to FALSE to receive all responses.
duration	The scan duration, in milliseconds. If set to zero or both duration and period set to non-zero, scanning will continue until DmScanStop() is called.
period	The scan period, in 1.28 sec units (only applicable to AE). If set to zero, periodic scanning is disabled.

Returns

None.

1.2.4.56 DmScanStop()

```
void DmScanStop (
            void  )
```

Stop scanning.

Returns

None.

1.2.4.57 DmScanSetInterval()

```
void DmScanSetInterval (
            uint8_t scanPhys,
            uint16_t * pScanInterval,
            uint16_t * pScanWindow )
```

Set the scan interval and window for the specified PHYs.

Parameters

scanPhys	Scanning PHYs.
pScanInterval	Scan interval array.
pScanWindow	Scan window array.

Returns

None.

1.2.4.58 DmScanSetAddrType()

```
void DmScanSetAddrType (
            uint8_t addrType )
```

Set the local address type used while scanning. This function can be used to configure scanning to use a random address.

Parameters

addrType	Address type.

Returns

None.

1.2.4.59 DmSyncStart()

```
dmSyncId_t DmSyncStart (
            uint8_t advSid,
            uint8_t advAddrType,
            const uint8_t * pAdvAddr,
            uint16_t skip,
            uint16_t syncTimeout )
```

Synchronize with periodic advertising from the given advertiser, and start receiving periodic advertising packets.

Note: The synchronization filter policy is used to determine whether the periodic advertiser list is used. If the periodic advertiser list is not used, the advertising SID, advertiser address type, and advertiser address parameters specify the periodic advertising device to listen to; otherwise these parameters are ignored.

Parameters

advSid	Advertising SID.
advAddrType	Advertiser address type.
pAdvAddr	Advertiser address.
skip	Number of periodic advertising packets that can be skipped after successful receive.
syncTimeout	Synchronization timeout.

Returns

Sync indentifier.

1.2.4.60 DmSyncStop()

```
void DmSyncStop (
            dmSyncId_t syncId )
```

Stop reception of the periodic advertising identified by the given sync identifier.

Parameters

syncId	Sync identifier.

Returns

None.

1.2.4.61 DmSyncSetEncrypt()

```
void DmSyncSetEncrypt (
            uint16_t syncHandle,
            bool_t encrypt )
```

Set the encryption mode of the Broadcast Isochronous Group (BIG) corresponding to the periodic advertising train identified by the sync handle.

Parameters

syncHandle	Synch handle.
encrypt	FALSE (Unencrypted) or FALSE (Encrypted).

Returns

None.

1.2.4.62 DmSyncEncrypted()

```
bool_t DmSyncEncrypted (
            uint16_t syncHandle )
```

Get the encryption mode of the Broadcast Isochronous Group (BIG) corresponding to the periodic advertising train identified by the sync handle.

Parameters

syncHandle	Synch handle.

Returns

TRUE if sync encrypted. FALSE, otherwise.

1.2.4.63 DmSyncEnabled()

```
bool_t DmSyncEnabled (
            uint16_t syncHandle )
```

Get status of sync identified by the handle.

Parameters

syncHandle	Synch handle.

Returns

TRUE if sync is enabled for that handle. FALSE, otherwise.

1.2.4.64 DmSyncInitialRptEnable()

```
void DmSyncInitialRptEnable (
            bool_t enable )
```

DM enable or disable initial periodic advertisement reporting.

Parameters

enable	TRUE to enable initial reporting, FALSE to disable initial reporting.

Returns

None.

1.2.4.65 DmBigSyncStart()

```
void DmBigSyncStart (
            uint8_t bigHandle,
            uint16_t syncHandle,
            uint8_t mse,
            uint16_t bigSyncTimeout,
            uint8_t numBis,
            uint8_t * pBis )
```

Synchronize to a Broadcast Isochronous Group (BIG) described in the periodic advertising train specified by the sync handle.

Parameters

bigHandle	BIG handle.
syncHandle	Periodic advertising train handle.
mse	Maximum number of subevents.
bigSyncTimeout	Synchronization timeout for the BIS, in the units of 10ms.
numBis	Total number of BISes in the BIG.
pBis	List of indices of BISes (in ascending order).

Returns

None.

1.2.4.66 DmBigSyncStop()

```
void DmBigSyncStop (
            uint8_t bigHandle )
```

Stop synchronizing or cancel the process of synchronizing to the Broadcast Isochronous Group (BIG) identified by the handle.

Note

The command also terminates the reception of BISes in the BIG specified in DmBigSyncStart, destroys the associated connection handles of the BISes in the BIG and removes the data paths for all BISes in the BIG.

Parameters

bigHandle	BIG handle.

Returns

None.

1.2.4.67 DmBisSyncInUse()

```
bool_t DmBisSyncInUse (
            uint16_t handle )
```

For internal use only. Return TRUE if the BIS sync is in use.

Parameters

handle	BIS connection handle.

Returns

TRUE if the BIS sync is in use, FALSE otherwise.

1.2.4.68 DmBigSyncSetBcastCode()

```
void DmBigSyncSetBcastCode (
            uint8_t bigHandle,
            bool_t encrypt,
            bool_t authen,
            uint8_t * pBcastCode )
```

Set the Broadcast Code for the given Broadcast Isochronous Group (BIG).

Parameters

bigHandle	BIG handle.
encrypt	FALSE (Unencrypted) or TRUE (Encrypted).
authen	FALSE (Unauthenticated) or TRUE (Authenticated).
pBcastCode	Broadcast code.

Returns

None.

1.2.4.69 DmBigSyncSetSecLevel()

```
void DmBigSyncSetSecLevel (
            uint8_t bigHandle,
            uint8_t secLevel )
```

Set the security level of the LE Security Mode 3 for the given Broadcast Isochronous Group (BIG).

Parameters

bigHandle	BIG handle.
secLevel	Security level.

Returns

None.

1.2.4.70 DmBigSyncGetSecLevel()

```
uint8_t DmBigSyncGetSecLevel (
            uint16_t handle )
```

Get the security level of the LE Security Mode 3 for the given Broadcast Isochronous Group (BIG) connection handle.

Parameters

handle	BIS connection handle.

Returns

Security level.

1.2.4.71 DmBisMasterInit()

```
void DmBisMasterInit (
            void  )
```

Initialize DM BIS manager for operation as master.

Returns

None.

1.2.4.72 DmAddDeviceToPerAdvList()

```
void DmAddDeviceToPerAdvList (
             uint8_t advAddrType,
             uint8_t * pAdvAddr,
             uint8_t advSid )
```

Add device to periodic advertiser list.

Parameters

advAddrType	Advertiser address type.
pAdvAddr	Advertiser address.
advSid	Advertising SID.

Returns

None.

1.2.4.73 DmRemoveDeviceFromPerAdvList()

```
void DmRemoveDeviceFromPerAdvList (
             uint8_t advAddrType,
             uint8_t * pAdvAddr,
             uint8_t advSid )
```

DM remove device from periodic advertiser list.

Parameters

advAddrType	Advertiser address type.
pAdvAddr	Advertiser address.
advSid	Advertising SID.

Returns

None.

1.2.4.74 DmClearPerAdvList()

```
void DmClearPerAdvList (
             void  )
```

DM clear periodic advertiser list.

Returns

None.

1.2.4.75 DmPastRptRcvEnable()

```
void DmPastRptRcvEnable (
            dmSyncId_t syncId,
            bool_t enable )
```

Enable or disable reports for the periodic advertising identified by the sync id.

Parameters

syncId	Sync identifier.
enable	TRUE to enable reporting, FALSE to disable reporting.

Returns

None.

1.2.4.76 DmPastSyncTrsf()

```
void DmPastSyncTrsf (
            dmConnId_t connId,
            uint16_t serviceData,
            dmSyncId_t syncId )
```

Send synchronization information about the periodic advertising identified by the sync id to a connected device.

Parameters

connId	Connection identifier.
serviceData	Value provided by the Host.
syncId	Sync identifier.

Returns

None.

1.2.4.77 DmPastSetInfoTrsf()

```
void DmPastSetInfoTrsf (
            dmConnId_t connId,
            uint16_t serviceData,
            uint8_t advHandle )
```

Send synchronization information about the periodic advertising in an advertising set to a connected device.

Parameters

connId	Connection identifier.
serviceData	Value provided by the Host.
advHandle	Advertising handle.

Returns

None.

1.2.4.78 DmPastConfig()

```
void DmPastConfig (
              dmConnId_t connId,
              uint8_t mode,
              uint16_t skip,
              uint16_t syncTimeout,
              uint8_t cteType )
```

Specify how the Controller should process periodic advertising synchronization information received from the device identified by the connnection handle.

Parameters

connId	Connection identifier.
mode	Action to be taken when periodic advertising info is received.
skip	Number of consecutive periodic advertising packets that the receiver may skip after successfully receiving a periodic advertising packet.
syncTimeout	Maximum permitted time between successful receives. If this time is exceeded, synchronization is lost.
cteType	Whether to only synchronize to periodic advertising with certain types of Constant Tone Extension.

Returns

None.

1.2.4.79 DmPastDefaultConfig()

```
void DmPastDefaultConfig (
              uint8_t mode,
              uint16_t skip,
              uint16_t syncTimeout,
              uint8_t cteType )
```

Specify the initial value for the mode, skip, timeout, and Constant Tone Extension type to be used for all subsequent connections over the LE transport.

Parameters

mode	Action to be taken when periodic advertising info is received.
skip	Number of consecutive periodic advertising packets that the receiver may skip after successfully receiving a periodic advertising packet.
syncTimeout	Maximum permitted time between successful receives. If this time is exceeded, synchronization is lost.
cteType	Whether to only synchronize to periodic advertising with certain types of Constant Tone Extension.

Returns

None.

1.2.4.80 DmConnCteRxSampleStart()

```
void DmConnCteRxSampleStart (
             dmConnId_t connId,
             uint8_t slotDurations,
             uint8_t switchPatternLen,
             uint8_t * pAntennaIDs )
```

Enable sampling received CTE fields on the specified connection, and configure the antenna switching pattern, and switching and sampling slot durations to be used.

Parameters

connId	Connection identifier.
slotDurations	Switching and sampling slot durations to be used while receiving CTE.
switchPatternLen	Number of Antenna IDs in switching pattern.
pAntennaIDs	List of Antenna IDs in switching pattern.

Returns

None.

1.2.4.81 DmConnCteRxSampleStop()

```
void DmConnCteRxSampleStop (
             dmConnId_t connId )
```

Disable sampling received CTE fields on the specified connection.

Parameters

connId	Connection identifier.

Returns

None.

1.2.4.82 DmConnCteTxConfig()

```
void DmConnCteTxConfig (
            dmConnId_t connId,
            uint8_t cteTypeBits,
            uint8_t switchPatternLen,
            uint8_t * pAntennaIDs )
```

Configure the antenna switching pattern, and permitted CTE types used for transmitting CTEs requested by the peer device on the specified connection.

Parameters

connId	Connection identifier.
cteTypeBits	Permitted CTE type bits used for transmitting CTEs requested by peer.
switchPatternLen	Number of Antenna IDs in switching pattern.
pAntennaIDs	List of Antenna IDs in switching pattern.

Returns

None.

1.2.4.83 DmConnCteReqStart()

```
void DmConnCteReqStart (
            dmConnId_t connId,
            uint16_t cteReqInt,
            uint8_t reqCteLen,
            uint8_t reqCteType )
```

Initiate the CTE Request procedure on the specified connection.

Parameters

connId	Connection identifier.
cteReqInt	CTE request interval.
reqCteLen	Minimum length of CTE being requested in 8 us units.
reqCteType	Requested CTE type.

Returns

None.

1.2.4.84 DmConnCteReqStop()

```
void DmConnCteReqStop (
            dmConnId_t connId )
```

Stop initiating the CTE Request procedure on the specified connection.

Parameters

connId	Connection identifier.

Returns

None.

1.2.4.85 DmConnCteRspStart()

```
void DmConnCteRspStart (
            dmConnId_t connId )
```

Start responding to LL_CTE_REQ PDUs with LL_CTE_RSP PDUs on the specified connection.

Parameters

connId	Connection identifier.

Returns

None.

1.2.4.86 DmConnCteRspStop()

```
void DmConnCteRspStop (
            dmConnId_t connId )
```

Stop responding to LL_CTE_REQ PDUs with LL_CTE_RSP PDUs on the specified connection.

Parameters

connId	Connection identifier.

Returns

None.

1.2.4.87 DmConnCteGetReqState()

```
uint8_t DmConnCteGetReqState (
            dmConnId_t connId )
```

Returns the device manager's CTE request state for a given connection.

Parameters

connId	Connection identifier.

Returns

The CTE request state.

1.2.4.88 DmConnCteGetRspState()

```
uint8_t DmConnCteGetRspState (
            dmConnId_t connId )
```

Returns the device manager's CTE response state for a given connection.

Parameters

connId	Connection identifier.

Returns

The CTE response state.

1.2.4.89 DmReadAntennaInfo()

```
void DmReadAntennaInfo (
            void  )
```

Read the switching rates, the sampling rates, the number of antennae, and the maximum length of a transmitted Constant Tone Extension supported by the Controller.

Returns

None.

Note

The antenna info will be returned with DM indication DM_READ_ANTENNA_INFO_IND.

1.2.4.90 DmConnInit()

```
void DmConnInit (
            void  )
```

Initialize DM connection manager.

Returns

None.

1.2.4.91 DmConnMasterInit()

```
void DmConnMasterInit (
            void  )
```

Initialize DM connection manager for operation as legacy master.

Returns

None.

1.2.4.92 DmExtConnMasterInit()

```
void DmExtConnMasterInit (
             void  )
```

Initialize DM connection manager for operation as extended master.

Returns

None.

1.2.4.93 DmConnSlaveInit()

```
void DmConnSlaveInit (
             void  )
```

Initialize DM connection manager for operation as legacy slave.

Returns

None.

1.2.4.94 DmExtConnSlaveInit()

```
void DmExtConnSlaveInit (
             void  )
```

Initialize DM connection manager for operation as extended slave.

Returns

None.

1.2.4.95 DmConnRegister()

```
void DmConnRegister (
             uint8_t clientId,
             dmCback_t cback )
```

Register with the DM connection manager.

Parameters

clientId	The client identifier.
cback	Client callback function.

Returns

None.

1.2.4.96 DmConnOpen()

```
dmConnId_t DmConnOpen (
            uint8_t clientId,
            uint8_t initPhys,
            uint8_t addrType,
            uint8_t * pAddr )
```

Open a connection to a peer device with the given address.

Parameters

clientId	The client identifier.
initPhys	Initiator PHYs.
addrType	Address type.
pAddr	Peer device address.

Returns

Connection identifier.

1.2.4.97 DmConnClose()

```
void DmConnClose (
            uint8_t clientId,
            dmConnId_t connId,
            uint8_t reason )
```

Close the connection with the give connection identifier.

Parameters

clientId	The client identifier.
connId	Connection identifier.
reason	Reason connection is being closed.

Returns

None.

1.2.4.98 DmConnAccept()

```
dmConnId_t DmConnAccept (
            uint8_t clientId,
            uint8_t advHandle,
            uint8_t advType,
            uint16_t duration,
            uint8_t maxEaEvents,
            uint8_t addrType,
            uint8_t * pAddr )
```

Accept a connection from the given peer device by initiating directed advertising.

Parameters

clientId	The client identifier.
advHandle	Advertising handle.
advType	Advertising type.
duration	Advertising duration (in ms).
maxEaEvents	Maximum number of extended advertising events.
addrType	Address type.
pAddr	Peer device address.

Returns

Connection identifier.

1.2.4.99 DmConnUpdate()

```
void DmConnUpdate (
            dmConnId_t connId,
            hciConnSpec_t * pConnSpec )
```

Update the connection parameters of an open connection.

Parameters

connId	Connection identifier.
pConnSpec	Connection specification.

Returns

None.

1.2.4.100 DmConnSetScanInterval()

```
void DmConnSetScanInterval (
            uint16_t scanInterval,
            uint16_t scanWindow )
```

Set the scan interval and window for connections to be created with DmConnOpen().

Parameters

scanInterval	The scan interval.
scanWindow	The scan window.

Returns

None.

1.2.4.101 DmExtConnSetScanInterval()

```
void DmExtConnSetScanInterval (
            uint8_t initPhys,
            uint16_t * pScanInterval,
            uint16_t * pScanWindow )
```

Set the scan interval and window for extended connections to be created with DmConnOpen().

Parameters

initPhys	Initiator PHYs.
pScanInterval	Scan interval array.
pScanWindow	Scan window array.

Returns

None.

1.2.4.102 DmConnSetConnSpec()

```
void DmConnSetConnSpec (
            hciConnSpec_t * pConnSpec )
```

Set the connection spec parameters for connections to be created with DmConnOpen().

Parameters

pConnSpec	Connection spec parameters.

Returns

None.

1.2.4.103 DmExtConnSetConnSpec()

```
void DmExtConnSetConnSpec (
             uint8_t initPhys,
             hciConnSpec_t * pConnSpec )
```

Set the extended connection spec parameters for extended connections to be created with DmConnOpen().

Parameters

initPhys	The initiator PHYs.
pConnSpec	Connection spec parameters array.

Returns

None.

1.2.4.104 DmConnSetAddrType()

```
void DmConnSetAddrType (
             uint8_t addrType )
```

Set the local address type used for connections created with DmConnOpen().

Parameters

addrType	Address type.

Returns

None.

1.2.4.105 DmConnSetIdle()

```
void DmConnSetIdle (
            dmConnId_t connId,
            uint16_t idleMask,
            uint8_t idle )
```

Configure a bit in the connection idle state mask as busy or idle.

Parameters

connId	Connection identifier.
idleMask	Bit in the idle state mask to configure.
idle	DM_CONN_BUSY or DM_CONN_IDLE.

Returns

None.

1.2.4.106 DmConnCheckIdle()

```
uint16_t DmConnCheckIdle (
            dmConnId_t connId )
```

Check if a connection is idle.

Parameters

connId	Connection identifier.

Returns

Zero if connection is idle, nonzero if busy.

1.2.4.107 DmConnReadRssi()

```
void DmConnReadRssi (
            dmConnId_t connId )
```

Read RSSI of a given connection.

Parameters

connId	Connection identifier.

Returns

None.

1.2.4.108 DmRemoteConnParamReqReply()

```
void DmRemoteConnParamReqReply (
            dmConnId_t connId,
            hciConnSpec_t * pConnSpec )
```

Reply to the HCI remote connection parameter request event. This command is used to indicate that the Host has accepted the remote device's request to change connection parameters.

Parameters

connId	Connection identifier.
pConnSpec	Connection specification.

Returns

None.

1.2.4.109 DmRemoteConnParamReqNegReply()

```
void DmRemoteConnParamReqNegReply (
            dmConnId_t connId,
            uint8_t reason )
```

Negative reply to the HCI remote connection parameter request event. This command is used to indicate that the Host has rejected the remote device's request to change connection parameters.

Parameters

connId	Connection identifier.
reason	Reason for rejection.

Returns

None.

1.2.4.110 DmConnSetDataLen()

```
void DmConnSetDataLen (
            dmConnId_t connId,
```

```
            uint16_t txOctets,
            uint16_t txTime )
```

Set data length for a given connection.

Parameters

connId	Connection identifier.
txOctets	Maximum number of payload octets for a Data PDU.
txTime	Maximum number of microseconds for a Data PDU.

Returns

None.

1.2.4.111 DmConnRole()

```
uint8_t DmConnRole (
            dmConnId_t connId )
```

Return the connection role indicating master or slave.

Parameters

connId	Connection identifier.

Returns

Device role.

1.2.4.112 DmWriteAuthPayloadTimeout()

```
void DmWriteAuthPayloadTimeout (
            dmConnId_t connId,
            uint16_t timeout )
```

Set authenticated payload timeout for a given connection.

Parameters

connId	Connection identifier.
timeout	Timeout period in units of 10ms.

Returns

None.

1.2.4.113 DmConnRequestPeerSca()

```
void DmConnRequestPeerSca (
            dmConnId_t connId )
```

Request the Sleep Clock Accuracy (SCA) of a peer device.

Parameters

connId	Connection identifier.

Returns

None.

1.2.4.114 DmCisInit()

```
void DmCisInit (
            void  )
```

Initialize DM Connected Isochronous Stream (CIS) manager.

Returns

None.

1.2.4.115 DmCisMasterInit()

```
void DmCisMasterInit (
            void  )
```

Initialize DM Connected Isochronous Stream (CIS) manager for operation as master.

Returns

None.

1.2.4.116 DmCisSlaveInit()

```
void DmCisSlaveInit (
            void  )
```

Initialize DM Connected Isochronous Stream (CIS) manager for operation as slave.

Returns

None.

1.2.4.117 DmCisCigSetSduInterval()

```
void DmCisCigSetSduInterval (
            uint8_t cigId,
            uint32_t sduIntervalMToS,
            uint32_t sduIntervalSToM )
```

Set the interval, in microseconds, of periodic SDUs for the given Connected Isochronous Group (CIG).

Parameters

cigId	CIG ID.
sduIntervalMToS	Time interval between start of consecutive SDUs from master Host.
sduIntervalSToM	Time interval between start of consecutive SDUs from slave Host.

Returns

None.

1.2.4.118 DmCisCigSetSca()

```
void DmCisCigSetSca (
            uint8_t cigId,
            uint8_t sca )
```

Set the slaves clock accuracy for the given Connected Isochronous Group (CIG).

Parameters

cigId	CIG identifier.
sca	Slaves clck accuracy (0 if unknown).

Returns

None.

Note

The slaves clock accuracy must which must be the worst-case sleep clock accuracy of the slaves that will participate in the CIG.

1.2.4.119 DmCisCigSetPackingFraming()

```
void DmCisCigSetPackingFraming (
            uint8_t cigId,
            uint8_t packing,
            uint8_t framing )
```

Set the packing scheme and framing format for the given Connected Isochronous Group (CIG).

Parameters

cigId	CIG identifier.
packing	Packing scheme.
framing	Indicates format of CIS Data PDUs.

Returns

None.

1.2.4.120 DmCisCigSetTransLatInterval()

```
void DmCisCigSetTransLatInterval (
            uint8_t cigId,
            uint16_t transLatMToS,
            uint16_t transLatSToM )
```

Set the maximum transport latency, in microseconds, for the given Connected Isochronous Group (CIG).

Parameters

cigId	CIG identifier.
transLatMToS	Maximum time for SDU to be transported from master Controller to slave Controller.
transLatSToM	Maximum time for SDU to be transported from slave Controller to master Controller.

Returns

None.

1.2.4.121 DmCisCigConfig()

```
void DmCisCigConfig (
            uint8_t cigId,
            dmConnId_t numCis,
            HciCisCisParams_t * pCisParam )
```

Set the parameters of one or more Connected Isochronous Streams (CISes) that are associated with the given Connected Isochronous Group (CIG).

Parameters

cigId	CIG identifier.
numCis	Number of CIS to be configured.
pCisParam	CIS parameters.

Returns

None.

1.2.4.122 DmCisCigRemove()

```
void DmCisCigRemove (
            uint8_t cigId )
```

Remove all the Connected Isochronous Streams (CISes) associated with the given Connected Isochronous Group (CIG).

Parameters

cigId	CIG identifier.

Returns

None.

1.2.4.123 DmCisOpen()

```
void DmCisOpen (
            uint8_t numCis,
            uint16_t * pCisHandle,
            dmConnId_t * pConnId )
```

Create one or more Connected Isochronous Streams (CISes) using the connections identified by the ACL connection handles.

Parameters

numCis	Total number of CISes to be created.
pCisHandle	List of connection handles of CISes.
pConnId	List of DM connection identifiers.

Returns

None.

1.2.4.124 DmCisAccept()

```
void DmCisAccept (
            uint16_t handle )
```

Inform the Controller to accept the request for the Connected Isochronous Stream (CIS) that is identified by the connection handle.

Parameters

handle	Connection handle of the CIS.

Returns

None.

1.2.4.125 DmCisReject()

```
void DmCisReject (
            uint16_t handle,
            uint8_t reason )
```

Inform the Controller to reject the request for the Connected Isochronous Stream (CIS) that is identified by the connection handle.

Parameters

handle	Connection handle of the CIS to be rejected.
reason	Reason the CIS request was rejected.

Returns

None.

1.2.4.126 DmCisClose()

```
void DmCisClose (
            uint16_t handle,
            uint8_t reason )
```

Close the Connected Isochronous Stream (CIS) connection with the given handle.

Parameters

handle	CIS connection handle.
reason	Reason connection is being closed.

Returns

None.

1.2.4.127 DmCisIdByHandle()

```
uint8_t DmCisIdByHandle (
            uint16_t handle )
```

For internal use only. Find the Connected Isochronous Stream (CIS) ID with matching handle.

Parameters

handle	CIS connection handle.

Returns

CIS identifier or DM_CIS_ID_NONE if error.

1.2.4.128 DmCisHandleById()

```
uint16_t DmCisHandleById (
             uint8_t cigId,
             uint8_t cisId )
```

For internal use only. Find the Connected Isochronous Stream (CIS) handle with matching CIG and CIS identifiers.

Parameters

handle	CIG ID.
handle	CIS ID.

Returns

CIS connection handle or DM_CONN_HCI_HANDLE_NONE if error.

1.2.4.129 DmCisConnInUse()

```
bool_t DmCisConnInUse (
             uint16_t handle )
```

For internal use only. Return TRUE if the Connected Isochronous Stream (CIS) connection is in use.

Parameters

handle	CIS connection handle.

Returns

TRUE if the CIS connection is in use, FALSE otherwise.

1.2.4.130 DmCisConnRole()

```
uint8_t DmCisConnRole (
             uint16_t handle )
```

For internal use only. Return the CIS connection role indicating master or slave.

Parameters

handle	CIS connection handle.

Returns

CIS connection role.

1.2.4.131 DmCisCigInUse()

```
bool_t DmCisCigInUse (
            uint8_t cigId )
```

For internal use only. Return TRUE if Connected Isochronous Group (CIG) is in use.

Parameters

cig↩ Id	CIG identifier.

Returns

TRUE if CIG is in use, FALSE otherwise.

1.2.4.132 DmCisInUse()

```
bool_t DmCisInUse (
            uint8_t cigId,
            uint8_t cisId )
```

For internal use only. Return TRUE if the Connected Isochronous Stream (CIS) connection is in use.

Parameters

cig↩ Id	CIG identifier.
cis↩ Id	CIS identifier.

Returns

TRUE if the CIS connection is in use, FALSE otherwise.

1.2.4.133 DmBisSlaveInit()

```
void DmBisSlaveInit (
            void  )
```

Initialize DM BIS manager for operation as slave.

Returns

None.

1.2.4.134 DmBigStart()

```
void DmBigStart (
            uint8_t bigHandle,
            uint8_t advHandle,
            uint8_t numBis,
            uint32_t sduInterUsec,
            uint16_t maxSdu,
            uint16_t mtlMs,
            uint8_t rtn )
```

Start a Broadcast Isochronous Group (BIG) with one or more Broadcast Isochronous Streams (BISes).

Parameters

bigHandle	CIG identifier.
advHandle	Used to identify the periodic advertising train.
numBis;	Total number of BISes in the BIG.
sduInterUsec	Interval, in microseconds, of BIG SDUs.
maxSdu	Maximum size of SDU
mtlMs	Maximum time, in milliseconds, for transmitting SDU.
rtn	Retransmitted number.

Returns

None.

1.2.4.135 DmBigStop()

```
void DmBigStop (
            uint8_t bigHandle,
            uint8_t reason )
```

Stop a Broadcast Isochronous Group (BIG) identified for the given handle.

Parameters

bigHandle	BIG identifier.
reason	Reason BIG is terminated.

Returns

None.

1.2.4.136 DmBisInUse()

```
bool_t DmBisInUse (
            uint16_t handle )
```

For internal use only. Return TRUE if the BIS is in use.

Parameters

handle	BIS connection handle.

Returns

TRUE if the BIS connection is in use, FALSE otherwise.

1.2.4.137 DmBigSetPhy()

```
void DmBigSetPhy (
            uint8_t bigHandle,
            uint8_t phyBits )
```

Set the PHYs used for transmission of PDUs of Broadcast Isochronous Streams (BISes) in Broadcast Isochronous Group (BIG).

Parameters

bigHandle	BIG handle.
phyBits	PHY bit field.

Returns

None.

1.2.4.138 DmBigSetPackingFraming()

```
void DmBigSetPackingFraming (
            uint8_t bigHandle,
            uint8_t packing,
            uint32_t framing )
```

Set the packing scheme and framing format for the given Broadcast Isochronous Group (BIG).

Parameters

bigHandle	BIG handle.
packing	Packing scheme.
framing	Indicates format of BIS Data PDUs.

Returns

None.

1.2.4.139 DmBigSetBcastCode()

```
void DmBigSetBcastCode (
             uint8_t bigHandle,
             bool_t encrypt,
             bool_t authen,
             uint8_t * pBcastCode )
```

Set the Broadcast Code for the given Broadcast Isochronous Group (BIG).

Parameters

bigHandle	BIG handle.
encrypt	FALSE (Unencrypted) or TRUE (Encrypted).
authen	FALSE (Unauthenticated) or TRUE (Authenticated).
pBcastCode	Broadcast code.

Returns

None.

1.2.4.140 DmBigSetSecLevel()

```
void DmBigSetSecLevel (
             uint8_t bigHandle,
             uint8_t secLevel )
```

Set the security level of the LE Security Mode 3 for the given Broadcast Isochronous Group (BIG).

Parameters

bigHandle	BIG handle.
secLevel	Security level.

Returns

None.

1.2.4.141 DmBigGetSecLevel()

```
uint8_t DmBigGetSecLevel (
            uint16_t handle )
```

Get the security level of the LE Security Mode 3 for the given Broadcast Isochronous Group (BIG) connection handle.

Parameters

handle	BIS connection handle.

Returns

Security level.

1.2.4.142 DmIsoInit()

```
void DmIsoInit (
            void  )
```

Initialize DM ISO manager.

Returns

None.

1.2.4.143 DmIsoRegister()

```
void DmIsoRegister (
            hciIsoCback_t cisCback,
            hciIsoCback_t bisCback )
```

Register CIS and BIS callbacks for the HCI ISO data path.

Parameters

cisCback	CIS data callback function.
bisCback	BIS data callback function.

Returns

None.

1.2.4.144 DmIsoDataPathSetup()

```
void DmIsoDataPathSetup (
            HciIsoSetupDataPath_t * pDataPathParam )
```

Setup the isochronous data path between the Host and the Controller for an established Connected Isochronous Stream (CIS) or Broadcast Isochronous Stream (BIS) identified by the connection handle parameter.

Parameters

pDataPathParam	Parameters to setup ISO data path.

Returns

None.

1.2.4.145 DmIsoDataPathRemove()

```
void DmIsoDataPathRemove (
            uint16_t handle,
            uint8_t directionBits )
```

Remove the input and/or output data path(s) associated with a Connected Isochronous Stream (CIS) or Broadcast Isochronous Stream (BIS) identified by the connection handle parameter.

Parameters

handle	Connection handle of CIS or BIS.
directionBits	Data path direction bits.

Returns

None.

1.2.4.146 DmDataPathConfig()

```
void DmDataPathConfig (
            HciConfigDataPath_t * pDataPathParam )
```

Request the Controller to configure the data transport path in a given direction between the Controller and the Host.

Parameters

pDataPathParam	Parameters for configuring data path.

Returns

None.

1.2.4.147 DmReadLocalSupCodecs()

```
void DmReadLocalSupCodecs (
            void  )
```

Read a list of the codecs supported by the Controller, as well as vendor specific codecs, which are defined by an individual manufacturer.

Returns

None.

1.2.4.148 DmReadLocalSupCodecCap()

```
void DmReadLocalSupCodecCap (
            HciReadLocalSupCodecCaps_t * pCodecParam )
```

Read a list of codec capabilities supported by the Controller for a given codec.

Parameters

pCodecParam	Parameters for reading local supported codec capabilities.

Returns

None.

1.2.4.149 DmReadLocalSupCtrDly()

```
void DmReadLocalSupCtrDly (
            HciReadLocalSupControllerDly_t * pDelayParam )
```

Read the range of supported Controller delays for the codec specified by Codec ID on a given transport type specified by Logical Transport Type, in the direction specified by Direction, and with the codec configuration specified by Codec Configuration.

Parameters

pDelayParam	Parameters for reading local supported controller delay.

Returns

None.

1.2.4.150 DmSendIsoData()

```
void DmSendIsoData (
            uint16_t handle,
            uint16_t len,
            uint8_t * pData )
```

Send ISO Data packet.

Parameters

pIsoParam	ISO data packet parameters.

1.2.4.151 DmSetDefaultPhy()

```
void DmSetDefaultPhy (
            uint8_t allPhys,
            uint8_t txPhys,
            uint8_t rxPhys )
```

Set the preferred values for the transmitter PHY and receiver PHY for all subsequent connections.

Parameters

allPhys	All PHYs preferences.
txPhys	Preferred transmitter PHYs.
rxPhys	Preferred receiver PHYs.

Returns

None.

1.2.4.152 DmReadPhy()

```
void DmReadPhy (
            dmConnId_t connId )
```

Read the current transmitter PHY and receiver PHY for a given connection.

Parameters

connId	Connection identifier.

Returns

None.

1.2.4.153 DmSetPhy()

```
void DmSetPhy (
            dmConnId_t connId,
            uint8_t allPhys,
            uint8_t txPhys,
            uint8_t rxPhys,
            uint16_t phyOptions )
```

Set the PHY preferences for a given connection.

Parameters

connId	Connection identifier.
allPhys	All PHYs preferences.
txPhys	Preferred transmitter PHYs.
rxPhys	Preferred receiver PHYs.
phyOptions	PHY options.

Returns

None.

1.2.4.154 DmPhyInit()

```
void DmPhyInit (
            void  )
```

Initialize DM PHY.

Returns

None.

1.2.4.155 DmDevReset()

```
void DmDevReset (
             void  )
```

Reset the device.

Returns

None.

1.2.4.156 DmDevSetRandAddr()

```
void DmDevSetRandAddr (
             uint8_t * pAddr )
```

Set the random address to be used by the local device.

Parameters

pAddr	Random address.

Returns

None.

1.2.4.157 DmDevWhiteListAdd()

```
void DmDevWhiteListAdd (
             uint8_t addrType,
             uint8_t * pAddr )
```

Add a peer device to the white list. Note that this function cannot be called while advertising, scanning, or connecting with white list filtering active.

Parameters

addrType	Address type.
pAddr	Peer device address.

Returns

None.

1.2.4.158 DmDevWhiteListRemove()

```
void DmDevWhiteListRemove (
             uint8_t addrType,
             uint8_t * pAddr )
```

Remove a peer device from the white list. Note that this function cannot be called while advertising, scanning, or connecting with white list filtering active.

Parameters

addrType	Address type.
pAddr	Peer device address.

Returns

None.

1.2.4.159 DmDevWhiteListClear()

```
void DmDevWhiteListClear (
             void  )
```

Clear the white list. Note that this function cannot be called while advertising, scanning, or connecting with white list filtering active.

Returns

None.

1.2.4.160 DmDevSetFilterPolicy()

```
bool_t DmDevSetFilterPolicy (
             uint8_t mode,
             uint8_t policy )
```

Set the Advertising, Scanning or Initiator filter policy.

Parameters

mode	Policy mode.
policy	Filter policy.

Returns

TRUE if the filter policy was successfully set, FALSE otherwise.

1.2.4.161 DmDevSetExtFilterPolicy()

```
bool_t DmDevSetExtFilterPolicy (
            uint8_t advHandle,
            uint8_t mode,
            uint8_t policy )
```

Set the Advertising filter policy for the given advertising, Scanning or Initiator filter policy.

Parameters

advHandle	Advertising handle (only applicable to advertising).
mode	Policy mode.
policy	Filter policy.

Returns

TRUE if the filter policy was successfully set, FALSE otherwise.

1.2.4.162 DmDevVsInit()

```
void DmDevVsInit (
            uint8_t param )
```

Vendor-specific controller initialization function.

Parameters

param	Vendor-specific parameter.

Returns

None.

1.2.4.163 DmSecInit()

```
void DmSecInit (
             void  )
```

Initialize DM security.

Returns

None.

1.2.4.164 DmSecLescInit()

```
void DmSecLescInit (
             void  )
```

Initialize DM LE Secure Connections security.

Returns

None.

1.2.4.165 DmSecPairReq()

```
void DmSecPairReq (
             dmConnId_t connId,
             uint8_t oob,
             uint8_t auth,
             uint8_t iKeyDist,
             uint8_t rKeyDist )
```

This function is called by a master device to initiate pairing.

Parameters

connId	DM connection ID.
oob	Out-of-band pairing data present or not present.
auth	Authentication and bonding flags.
iKeyDist	Initiator key distribution flags.
rKeyDist	Responder key distribution flags.

Returns

None.

1.2.4.166 DmSecPairRsp()

```
void DmSecPairRsp (
            dmConnId_t connId,
            uint8_t oob,
            uint8_t auth,
            uint8_t iKeyDist,
            uint8_t rKeyDist )
```

This function is called by a slave device to proceed with pairing after a DM_SEC_PAIR_IND event is received.

Parameters

connId	DM connection ID.
oob	Out-of-band pairing data present or not present.
auth	Authentication and bonding flags.
iKeyDist	Initiator key distribution flags.
rKeyDist	Responder key distribution flags.

Returns

None.

1.2.4.167 DmSecCancelReq()

```
void DmSecCancelReq (
            dmConnId_t connId,
            uint8_t reason )
```

This function is called to cancel the pairing process.

Parameters

connId	DM connection ID.
reason	Failure reason.

Returns

None.

1.2.4.168 DmSecAuthRsp()

```
void DmSecAuthRsp (
            dmConnId_t connId,
```

```
            uint8_t authDataLen,
            uint8_t * pAuthData )
```

This function is called in response to a DM_SEC_AUTH_REQ_IND event to provide PIN or OOB data during pairing.

Parameters

connId	DM connection ID.
authDataLen	Length of PIN or OOB data.
pAuthData	pointer to PIN or OOB data.

Returns

None.

1.2.4.169 DmSecSlaveReq()

```
void DmSecSlaveReq (
              dmConnId_t connId,
              uint8_t auth )
```

This function is called by a slave device to request that the master initiates pairing or link encryption.

Parameters

connId	DM connection ID.
auth	Authentication flags.

Returns

None.

1.2.4.170 DmSecEncryptReq()

```
void DmSecEncryptReq (
              dmConnId_t connId,
              uint8_t secLevel,
              dmSecLtk_t * pLtk )
```

This function is called by a master device to initiate link encryption.

Parameters

connId	DM connection ID.
secLevel	Security level of pairing when LTK was exchanged.
pLtk	Pointer to LTK parameter structure.

Returns

None.

1.2.4.171 DmSecLtkRsp()

```
void DmSecLtkRsp (
            dmConnId_t connId,
            bool_t keyFound,
            uint8_t secLevel,
            uint8_t * pKey )
```

This function is called by a slave in response to a DM_SEC_LTK_REQ_IND event to provide the long term key used for encryption.

Parameters

connId	DM connection ID.
keyFound	TRUE if key found.
secLevel	Security level of pairing when key was exchanged.
pKey	Pointer to the key, if found.

Returns

None.

1.2.4.172 DmSecSetLocalCsrk()

```
void DmSecSetLocalCsrk (
            uint8_t * pCsrk )
```

This function sets the local CSRK used by the device.

Parameters

pCsrk	Pointer to CSRK.

Returns

None.

1.2.4.173 DmSecSetLocalIrk()

```
void DmSecSetLocalIrk (
             uint8_t * pIrk )
```

This function sets the local IRK used by the device.

Parameters

pIrk	Pointer to IRK.

Returns

None.

1.2.4.174 DmSecGenerateEccKeyReq()

```
void DmSecGenerateEccKeyReq (
             void  )
```

This function generates an ECC key for use with LESC security.

Returns

None.

1.2.4.175 DmSecSetEccKey()

```
void DmSecSetEccKey (
             secEccKey_t * pKey )
```

This function sets the ECC key for use with LESC security.

Parameters

pKey	Pointer to key.

Returns

None.

1.2.4.176 DmSecGetEccKey()

```
secEccKey_t* DmSecGetEccKey (
             void  )
```

This function gets the local ECC key for use with LESC security.

Returns

Pointer to local ECC key.

1.2.4.177 DmSecSetDebugEccKey()

```
void DmSecSetDebugEccKey (
             void  )
```

This function sets the ECC key for use with LESC security to standard debug keys values.

Returns

None.

1.2.4.178 DmSecSetOob()

```
void DmSecSetOob (
             dmConnId_t connId,
             dmSecLescOobCfg_t * pConfig )
```

This function configures the DM to use OOB pairing for the given connection. The pRand and pConfirm contain the Random and Confirm values exchanged via out of band methods.

Parameters

connId	ID of the connection.
pConfig	Pointer to OOB configuration.

Returns

Pointer to IRK.

1.2.4.179 DmSecCalcOobReq()

```
void DmSecCalcOobReq (
             uint8_t * pRand,
             uint8_t * pPubKeyX )
```

This function calculates the local random and confirm values used in LESC OOB pairing. The operation's result is posted as a DM_SEC_CALC_OOB_IND event to the application's DM callback handler. The local rand and confirm values are exchanged with the peer via out-of-band (OOB) methods and passed into the DmSecSetOob after DM_CONN_OPEN_IND.

Parameters

pRand	Random value used in calculation.
pPubKeyX	X component of the local public key.

Returns

None.

1.2.4.180 DmSecCompareRsp()

```
void DmSecCompareRsp (
            dmConnId_t connId,
            bool_t valid )
```

This function is called by the application in response to a DM_SEC_COMPARE_IND event. The valid parameter indicates if the compare value of the DM_SEC_COMPARE_IND was valid.

Parameters

connId	ID of the connection.
valid	TRUE if compare value was valid

Returns

None.

1.2.4.181 DmSecGetCompareValue()

```
uint32_t DmSecGetCompareValue (
            uint8_t * pConfirm )
```

This function returns the 6-digit compare value for the specified 128-bit confirm value.

Parameters

pConfirm	Pointer to 128-bit comfirm value.

Returns

Six-digit compare value.

1.2.4.182 DmLlAddrType()

```
uint8_t DmLlAddrType (
            uint8_t addrType )
```

Map an address type to a type used by LL.

Parameters

addrType	Address type used by Host.

Returns

Address type used by LL.

1.2.4.183 DmHostAddrType()

```
uint8_t DmHostAddrType (
            uint8_t addrType )
```

Map an address type to a type used by Host.

Parameters

addrType	Address type used by LL.

Returns

Address type used by Host.

1.2.4.184 DmSizeOfEvt()

```
uint16_t DmSizeOfEvt (
            dmEvt_t * pDmEvt )
```

Return size of a DM callback event.

Parameters

pDmEvt	DM callback event.

Returns

Size of DM callback event.

1.2.4.185 DmL2cConnUpdateCnf()

```
void DmL2cConnUpdateCnf (
             uint16_t handle,
             uint16_t reason )
```

For internal use only. L2C calls this function to send the result of an L2CAP connection update response to DM.

Parameters

handle	Connection handle.
reason	Connection update response reason code.

Returns

None.

1.2.4.186 DmL2cCmdRejInd()

```
void DmL2cCmdRejInd (
             uint16_t handle,
             uint16_t result )
```

For internal use only. L2C calls this function to send the result of an L2CAP Command Reject up to the application.

Parameters

handle	Connection handle.
result	Connection update result code.

Returns

None.

1.2.4.187 DmL2cConnUpdateInd()

```
void DmL2cConnUpdateInd (
             uint8_t identifier,
             uint16_t handle,
             hciConnSpec_t * pConnSpec )
```

For internal use only. L2C calls this function when it receives a connection update request from a peer device.

Parameters

identifier	Identifier value.
handle	Connection handle.
pConnSpec	Connection spec parameters.

Returns

None.

1.2.4.188 DmConnIdByHandle()

```
dmConnId_t DmConnIdByHandle (
             uint16_t handle )
```

For internal use only. Find the connection ID with matching handle.

Parameters

handle	Handle to find.

Returns

Connection ID or DM_CONN_ID_NONE if error.

1.2.4.189 DmConnInUse()

```
bool_t DmConnInUse (
             dmConnId_t connId )
```

For internal use only. Return TRUE if the connection is in use.

Parameters

connId	Connection ID.

Returns

TRUE if the connection is in use, FALSE otherwise.

1.2.4.190 DmConnActiveCount()

```
uint8_t DmConnActiveCount (
            void  )
```

Count active connections *.

Returns

Number of active connections.

1.2.4.191 DmConnPeerAddrType()

```
uint8_t DmConnPeerAddrType (
            dmConnId_t connId )
```

For internal use only. Return the peer address type.

Parameters

connId	Connection ID.

Returns

Peer address type.

1.2.4.192 DmConnPeerAddr()

```
uint8_t* DmConnPeerAddr (
            dmConnId_t connId )
```

For internal use only. Return the peer device address.

Parameters

connId	Connection ID.

Returns

Pointer to peer device address.

1.2.4.193 DmConnLocalAddrType()

```
uint8_t DmConnLocalAddrType (
            dmConnId_t connId )
```

For internal use only. Return the local address type.

Parameters

connId	Connection ID.

Returns

Local address type.

1.2.4.194 DmConnLocalAddr()

```
uint8_t* DmConnLocalAddr (
            dmConnId_t connId )
```

For internal use only. Return the local address.

Parameters

connId	Connection ID.

Returns

Pointer to local address.

1.2.4.195 DmConnPeerRpa()

```
uint8_t* DmConnPeerRpa (
            dmConnId_t connId )
```

For internal use only. Return the peer resolvable private address (RPA).

Parameters

connId	Connection ID.

Returns

Pointer to peer RPA.

1.2.4.196 DmConnLocalRpa()

```
uint8_t* DmConnLocalRpa (
            dmConnId_t connId )
```

For internal use only. Return the local resolvable private address (RPA).

Parameters

connId	Connection ID.

Returns

Pointer to local RPA.

1.2.4.197 DmConnSecLevel()

```
uint8_t DmConnSecLevel (
            dmConnId_t connId )
```

For internal use only. Return the security level of the connection.

Parameters

connId	Connection ID.

Returns

Security level of the connection.

1.2.4.198 DmSmpEncryptReq()

```
void DmSmpEncryptReq (
             dmConnId_t connId,
             uint8_t secLevel,
             uint8_t * pKey )
```

For internal use only. This function is called by SMP to request encryption.

Parameters

connId	DM connection ID.
secLevel	Security level of pairing when key was exchanged.
pKey	Pointer to key.

Returns

None.

1.2.4.199 DmSmpCbackExec()

```
void DmSmpCbackExec (
             dmEvt_t * pDmEvt )
```

For internal use only. Execute DM callback from SMP procedures.

Parameters

pDmEvt	Pointer to callback event data.

Returns

None.

1.2.4.200 DmSecGetLocalCsrk()

```
uint8_t* DmSecGetLocalCsrk (
             void  )
```

For internal use only. This function gets the local CSRK used by the device.

Returns

Pointer to CSRK.

1.2.4.201 DmSecGetLocalIrk()

```
uint8_t* DmSecGetLocalIrk (
              void  )
```

For internal use only. This function gets the local IRK used by the device.

Returns

Pointer to IRK.

1.2.4.202 DmReadRemoteFeatures()

```
void DmReadRemoteFeatures (
              dmConnId_t connId )
```

For internal use only. Read the features of the remote device.

Parameters

connId	Connection identifier.

Returns

None.

1.2.4.203 DmReadRemoteVerInfo()

```
void DmReadRemoteVerInfo (
              dmConnId_t connId )
```

Read the version info of the remote device.

Parameters

connId	Connection identifier.

Returns

None.

1.2.4.204 DmDisableSlaveLatency()

```
void DmDisableSlaveLatency (
            dmConnId_t connId,
            bool_t disabled )
```

Disable Slave Latency.

Parameters

connId	Connection identifier.
disabled	disable/enable

Returns

None.

1.2.4.205 DmOverruleRemoteMaxRxOctetsAndTime()

```
void DmOverruleRemoteMaxRxOctetsAndTime (
            dmConnId_t connId,
            uint16_t maxRxOctetsRemote,
            uint16_t maxRxTimeRemote )
```

Over rule Remote Maximum Rx octets.

Parameters

connId	Connection identifier.
maxRxOctetsRemote	Remote maximum receive octets.
maxRxTimeRemote	Remote maximum Recieve Time.

Returns

None.

1.2.4.206 HciVsdSetDeviceAddress()

```
void HciVsdSetDeviceAddress (
            uint8_t * pAddr )
```

Set device address.

Parameters

pAddr	pointer to the address.

Returns

None.

1.2.4.207 HciVsdSetTransmitPower()

```
void HciVsdSetTransmitPower (
            int8_t transmitPower )
```

Set transmit power.

Parameters

transmitPower	TX Power value.

Returns

None.

1.2.4.208 HciCmndVsdSetLeMetaVSDEvent()

```
void HciCmndVsdSetLeMetaVSDEvent (
            uint8_t event )
```

Set event notification bit.

Parameters

event	bit to receive vsd notifications in the host stack.

Returns

None.

1.2.4.209 HciCmndVsdResetLeMetaVSDEvent()

```
void HciCmndVsdResetLeMetaVSDEvent (
            uint8_t event )
```

Reset event notification bit.

Parameters

event	bit to disable vsd notifications in the host stack.

Returns

None.

1.3 STACK_EVENT

DM Event Handling

Message passing interface to DM from other tasks through WSF.

- void DmHandlerInit (wsfHandlerId_t handlerId)

  DM handler init function called during system initialization.
- void DmHandler (wsfEventMask_t event, wsfMsgHdr_t *pMsg)

  WSF event handler for DM.

1.3.1 Detailed Description

1.3.2 Function Documentation

1.3.2.1 DmHandlerInit()

```
void DmHandlerInit (
            wsfHandlerId_t handlerId )
```

DM handler init function called during system initialization.

Parameters

handlerId	WSF handler ID for DM.

Returns

None.

1.3.2.2 DmHandler()

```
void DmHandler (
            wsfEventMask_t event,
            wsfMsgHdr_t * pMsg )
```

WSF event handler for DM.

Parameters

event	WSF event mask.
pMsg	WSF message.

Returns

None.

1.4 WSF_OS_API

Data Structures

- struct wsfMsgHdr_t

  Common message structure passed to event handler.

Macros

- #define WSF_OS_DIAG FALSE

  OS Diagnostics.
- #define WSF_TASK_FROM_ID(handlerID) (((handlerID) >> 4) & 0x0F)

  Derive task from handler ID.
- #define WSF_HANDLER_FROM_ID(handlerID) ((handlerID) & 0x0F)

  Derive handler from handler ID.
- #define WSF_INVALID_TASK_ID 0xFF

  Invalid Task Identifier.
- #define WSF_OS_GET_ACTIVE_HANDLER_ID() WSF_INVALID_TASK_ID

  Get Diagnostic Task Identifier.

Typedefs

- typedef uint8_t wsfHandlerId_t

  Event handler ID data type.
- typedef uint16_t wsfEventMask_t

  Event handler event mask data type.
- typedef wsfHandlerId_t wsfTaskId_t

  Task ID data type.
- typedef uint8_t wsfTaskEvent_t

  Task event mask data type.
- typedef bool_t(* WsfOsIdleHandler_t) (void)

  Idle check function.
- typedef void(* wsfEventHandler_t) (wsfEventMask_t event, wsfMsgHdr_t *pMsg)

  Event handler callback function.

Functions

- void WsfSetEvent (wsfHandlerId_t handlerId, wsfEventMask_t event)

  Set an event for an event handler.
- void WsfTaskLock (void)

  Lock task scheduling.
- void WsfTaskUnlock (void)

  Unlock task scheduling.
- void WsfTaskSetReady (wsfHandlerId_t handlerId, wsfTaskEvent_t event)

  Set the task used by the given handler as ready to run.
- wsfQueue_t * WsfTaskMsgQueue (wsfHandlerId_t handlerId)

  Return the task message queue used by the given handler.
- wsfHandlerId_t WsfOsSetNextHandler (wsfEventHandler_t handler)

Set the next WSF handler function in the WSF OS handler array. This function should only be called as part of the OS initialization procedure.

- void WsfOsInit (void)

  Initialize OS control structure.
- bool_t WsfOsReadyToSleep (void)

  Check if WSF is ready to sleep.
- void WsfOsDispatcher (void)

  Event dispatched. Designed to be called repeatedly from infinite loop.
- void WsfOsEnterMainLoop (void)

  OS starts main loop.
- void WsfOsRegisterIdleTask (WsfOsIdleHandler_t func)

  Register service check functions.

Variables

- wsfHandlerId_t WsfActiveHandler

  Diagnostic Task Identifier.

WSF Task Events

- #define WSF_MSG_QUEUE_EVENT 0x01

  Message queued for event handler.
- #define WSF_TIMER_EVENT 0x02

  Timer expired for event handler.
- #define WSF_HANDLER_EVENT 0x04

  Event set for event handler.

1.4.1 Detailed Description

1.4.2 Typedef Documentation

1.4.2.1 wsfEventHandler_t

```
typedef void(* wsfEventHandler_t) (wsfEventMask_t event, wsfMsgHdr_t *pMsg)
```

Event handler callback function.

Parameters

event	Mask of events set for the event handler.
pMsg	Pointer to message for the event handler.

Definition at line 151 of file wsf_os.h.

1.4.3 Function Documentation

1.4.3.1 WsfSetEvent()

```
void WsfSetEvent (
            wsfHandlerId_t handlerId,
            wsfEventMask_t event )
```

Set an event for an event handler.

Parameters

handlerId	Handler ID.
event	Event or events to set.

1.4.3.2 WsfTaskSetReady()

```
void WsfTaskSetReady (
            wsfHandlerId_t handlerId,
            wsfTaskEvent_t event )
```

Set the task used by the given handler as ready to run.

Parameters

handlerId	Event handler ID.
event	Task event mask.

1.4.3.3 WsfTaskMsgQueue()

```
wsfQueue_t* WsfTaskMsgQueue (
            wsfHandlerId_t handlerId )
```

Return the task message queue used by the given handler.

Parameters

handlerId	Event handler ID.

Returns

Task message queue.

1.4.3.4 WsfOsSetNextHandler()

```
wsfHandlerId_t WsfOsSetNextHandler (
            wsfEventHandler_t handler )
```

Set the next WSF handler function in the WSF OS handler array. This function should only be called as part of the OS initialization procedure.

Parameters

handler	WSF handler function.

Returns

WSF handler ID for this handler.

1.4.3.5 WsfOsInit()

```
void WsfOsInit (
            void  )
```

Initialize OS control structure.

Returns

None.

1.4.3.6 WsfOsReadyToSleep()

```
bool_t WsfOsReadyToSleep (
            void  )
```

Check if WSF is ready to sleep.

Returns

Return TRUE if there are no pending WSF task events set, FALSE otherwise.

1.4.3.7 WsfOsDispatcher()

```
void WsfOsDispatcher (
             void  )
```

Event dispatched. Designed to be called repeatedly from infinite loop.

Returns

None.

1.4.3.8 WsfOsRegisterIdleTask()

```
void WsfOsRegisterIdleTask (
             WsfOsIdleHandler_t func )
```

Register service check functions.

Parameters

func	Service function.

1.5 WSF_TYPES

Integer Data Types

- #define **bool_t** uint8_t
- #define **FALSE** 0
- #define **TRUE** (!FALSE)
- #define **UINT64_C**(x) x##ULL
- #define **UINT32_C**(x) x##UL
- #define **UINT8_C**(x) (x)

1.5.1 Detailed Description

Chapter 2

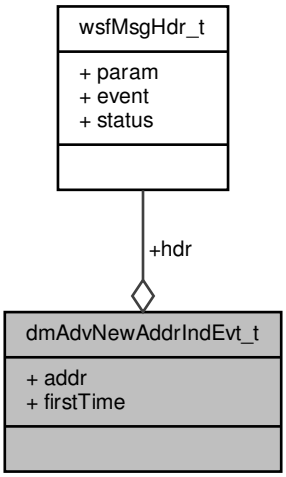
Data Structure Documentation

2.1 dmAdvNewAddrIndEvt_t Struct Reference

Data type for DM_ADV_NEW_ADDR_IND.

`#include <dm_api.h>`

Collaboration diagram for dmAdvNewAddrIndEvt_t:



Data Fields

- wsfMsgHdr_t hdr

  Header.
- bdAddr_t addr

  New resolvable private address.
- bool_t firstTime

  TRUE when address is generated for the first time.

2.1.1 Detailed Description

Data type for DM_ADV_NEW_ADDR_IND.

Definition at line 737 of file dm_api.h.

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

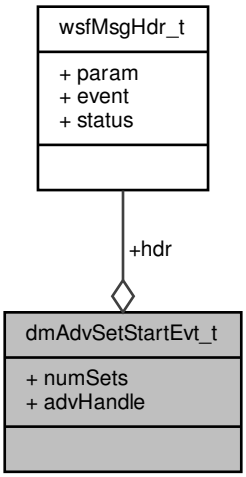
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.2 dmAdvSetStartEvt_t Struct Reference

Data structure for DM_ADV_SET_START_IND.

`#include <dm_api.h>`

Collaboration diagram for dmAdvSetStartEvt_t:



Data Fields

- wsfMsgHdr_t hdr

  Header.
- uint8_t numSets

  Number of advertising sets.
- uint8_t advHandle [DM_NUM_ADV_SETS]

  Advertising handle array.

2.2.1 Detailed Description

Data structure for DM_ADV_SET_START_IND.

Definition at line 745 of file dm_api.h.

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

- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.3 dmCfg_t Struct Reference

Configuration structure.

```
#include <dm_api.h>
```

Collaboration diagram for dmCfg_t:

Data Fields

- uint8_t dummy

  Placeholder variable.

2.3.1 Detailed Description

Configuration structure.

Definition at line 623 of file dm_api.h.

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

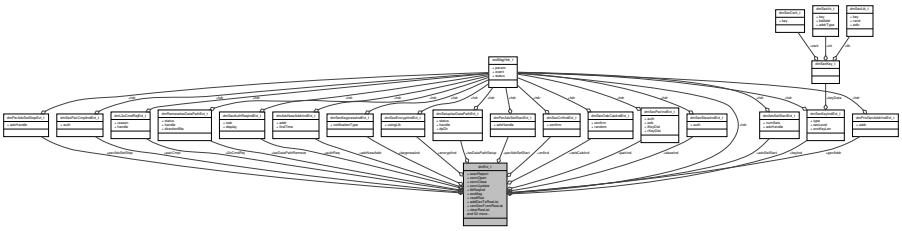
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.4 dmEvt_t Union Reference

Union of DM callback event data types.

`#include <dm_api.h>`

Collaboration diagram for dmEvt_t:



Data Fields

- wsfMsgHdr_t hdr
  Common header.
- dmAdvNewAddrIndEvt_t advNewAddr
  handles DM_ADV_NEW_ADDR_IND
- hciLeAdvReportEvt_t scanReport
  handles DM_SCAN_REPORT_IND
- hciLeConnCmplEvt_t connOpen
  handles DM_CONN_OPEN_IND
- hciDisconnectCmplEvt_t connClose
  handles DM_CONN_CLOSE_IND
- hciLeConnUpdateCmplEvt_t connUpdate
  handles DM_CONN_UPDATE_IND
- dmSecPairCmplIndEvt_t pairCmpl
  handles DM_SEC_PAIR_CMPL_IND
- dmSecEncryptIndEvt_t encryptInd
  handles DM_SEC_ENCRYPT_IND
- dmSecAuthReqIndEvt_t authReq
  handles DM_SEC_AUTH_REQ_IND
- dmSecKeyIndEvt_t keyInd
  handles DM_SEC_KEY_IND
- hciLeLtkReqEvt_t ltkReqInd
  handles DM_SEC_LTK_REQ_IND
- dmSecPairIndEvt_t pairInd
  handles DM_SEC_PAIR_IND
- dmSecSlaveIndEvt_t slaveInd
  handles DM_SEC_SLAVE_REQ_IND
- dmSecOobCalcIndEvt_t oobCalcInd
  handles DM_SEC_CALC_OOB_IND
- secEccMsg_t eccMsg
  handles DM_SEC_ECC_KEY_IND
- dmSecCnfIndEvt_t cnfInd

handles DM_SEC_COMPARE_IND

- dmSecKeypressIndEvt_t keypressInd

  handles DM_SEC_KEYPRESS_IND
- dmPrivGenAddrIndEvt_t genAddr

  handles DM_PRIV_GENERATE_ADDR_IND
- hciReadRssiCmdCmplEvt_t readRssi

  handles DM_CONN_READ_RSSI_IND
- hciLeAddDevToResListCmdCmplEvt_t addDevToResList

  handles DM_PRIV_ADD_DEV_TO_RES_LIST_IND
- hciLeRemDevFromResListCmdCmplEvt_t remDevFromResList

  handles DM_PRIV_REM_DEV_FROM_RES_LIST_IND
- hciLeClearResListCmdCmplEvt_t clearResList

  handles DM_PRIV_CLEAR_RES_LIST_IND
- hciLeReadPeerResAddrCmdCmplEvt_t readPeerResAddr

  handles DM_PRIV_READ_PEER_RES_ADDR_IND
- hciLeReadLocalResAddrCmdCmplEvt_t readLocalResAddr

  handles DM_PRIV_READ_LOCAL_RES_ADDR_IND
- hciLeSetAddrResEnableCmdCmplEvt_t setAddrResEnable

  handles DM_PRIV_SET_ADDR_RES_ENABLE_IND
- hciLeRemConnParamReqEvt_t remConnParamReq

  handles DM_REM_CONN_PARAM_REQ_IND
- hciLeDataLenChangeEvt_t dataLenChange

  handles DM_CONN_DATA_LEN_CHANGE_IND
- hciWriteAuthPayloadToCmdCmplEvt_t writeAuthTo

  handles DM_CONN_WRITE_AUTH_TO_IND
- hciAuthPayloadToExpiredEvt_t authToExpired

  handles DM_CONN_AUTH_TO_EXPIRED_IND
- hciLeReadPhyCmdCmplEvt_t readPhy

  handles DM_PHY_READ_IND
- hciLeSetDefPhyCmdCmplEvt_t setDefPhy

  handles DM_PHY_SET_DEF_IND
- hciLePhyUpdateEvt_t phyUpdate

  handles DM_PHY_UPDATE_IND
- dmAdvSetStartEvt_t advSetStart

  handles DM_ADV_SET_START_IND
- hciLeAdvSetTermEvt_t advSetStop

  handles DM_ADV_SET_STOP_IND
- hciLeScanReqRcvdEvt_t scanReqRcvd

  handles DM_SCAN_REQ_RCVD_IND
- hciLeExtAdvReportEvt_t extScanReport

  handles DM_EXT_SCAN_REPORT_IND
- dmPerAdvSetStartEvt_t perAdvSetStart

  handles DM_PER_ADV_SET_START_IND
- dmPerAdvSetStopEvt_t perAdvSetStop

  handles DM_PER_ADV_SET_STOP_IND
- hciLePerAdvSyncEstEvt_t perAdvSyncEst

  handles DM_PER_ADV_SYNC_EST_IND
- hciLePerAdvSyncEstEvt_t perAdvSyncEstFail

  handles DM_PER_ADV_SYNC_EST_FAIL_IND
- hciLePerAdvSyncLostEvt_t perAdvSyncLost

  handles DM_PER_ADV_SYNC_LOST_IND

- HciLePerAdvSyncTrsfRcvdEvt_t perAdvSyncTrsfEst
  handles DM_PER_ADV_SYNC_TRSF_EST_IND
- HciLePerAdvSyncTrsfRcvdEvt_t perAdvSyncTrsEstFail
  handles DM_PER_ADV_SYNC_TRSF_EST_FAIL_IND
- hciLePerAdvSyncTrsfCmdCmplEvt_t perAdvSyncTrsf
  handles DM_PER_ADV_SYNC_TRSF_IND
- hciLePerAdvSetInfoTrsfCmdCmplEvt_t perAdvSetInfoTrsf
  handles DM_PER_ADV_SET_INFO_TRSF_IND
- hciLePerAdvReportEvt_t perAdvReport
  handles DM_PER_ADV_REPORT_IND
- hciLeReadRemoteFeatCmplEvt_t readRemoteFeat
  handles DM_REMOTE_FEATURES_IND
- hciReadRemoteVerInfoCmplEvt_t readRemVerInfo
  handles DM_READ_REMOTE_VER_INFO_IND
- hciLeConnIQReportEvt_t connIQReport
  handles DM_CONN_IQ_REPORT_IND
- hciLeCteReqFailedEvt_t cteReqFail
  handles DM_CTE_REQ_FAIL_IND
- hciLeSetConnCteRxParamsCmdCmplEvt_t connCteRxSampleStart
  handles DM_CONN_CTE_RX_SAMPLE_START_IND
- hciLeSetConnCteRxParamsCmdCmplEvt_t connCteRxSampleStop
  handles DM_CONN_CTE_RX_SAMPLE_STOP_IND
- hciLeSetConnCteTxParamsCmdCmplEvt_t connCteTxCfg
  handles DM_CONN_CTE_TX_CFG_IND
- hciLeConnCteReqEnableCmdCmplEvt_t connCteReqStart
  handles DM_CONN_CTE_REQ_START_IND
- hciLeConnCteReqEnableCmdCmplEvt_t connCteReqStop
  handles DM_CONN_CTE_REQ_STOP_IND
- hciLeConnCteRspEnableCmdCmplEvt_t connCteRspStart
  handles DM_CONN_CTE_RSP_START_IND
- hciLeConnCteRspEnableCmdCmplEvt_t connCteRspStop
  handles DM_CONN_CTE_RSP_STOP_IND
- hciLeReadAntennaInfoCmdCmplEvt_t readAntennaInfo
  handles DM_READ_ANTENNA_INFO_IND
- hciLeSetCigParamsCmdCmplEvt_t cisCigConfig
  handles DM_CIS_CIG_CONFIG_IND
- hciLeRemoveCigCmdCmplEvt_t cisCigRemove
  handles DM_CIS_CIG_REMOVE_IND
- HciLeCisReqEvt_t cisReq
  handles DM_CIS_REQ_IND
- HciLeCisEstEvt_t cisOpen
  handles DM_CIS_OPEN_IND
- hciDisconnectCmplEvt_t cisClose
  handles DM_CIS_CLOSE_IND
- HciLeReqPeerScaCmplEvt_t_t reqPeerSca
  handles DM_REQ_PEER_SCA_IND
- dmSetupIsoDataPathEvt_t isoDataPathSetup
  handles DM_ISO_DATA_PATH_SETUP_IND
- dmRemoveIsoDataPathEvt_t isoDataPathRemove
  handles DM_ISO_DATA_PATH_REMOVE_IND
- hciConfigDataPathCmdCmplEvt_t dataPathConfig

handles DM_DATA_PATH_CONFIG_IND

- hciReadLocalSupCodecsCmdCmplEvt_t readLocalSupCodecs

  handles DM_READ_LOCAL_SUP_CODECS_IND
- hciReadLocalSupCodecCapCmdCmplEvt_t readLocalSupCodecCap

  handles DM_READ_LOCAL_SUP_CODEC_CAP_IND
- hciReadLocalSupCtrDlyCmdCmplEvt_t readLocalSupCtrDly

  handles DM_READ_LOCAL_SUP_CTR_DLY_IND
- HciLeCreateBigCmplEvt_t bigStart

  handles DM_BIG_START_IND
- HciLeTerminateBigCmplEvt_t bigStop

  handles DM_BIG_STOP_IND
- HciLeBigSyncEstEvt_t bigSyncEst

  handles DM_BIG_SYNC_EST_IND
- HciLeBigSyncEstEvt_t bigSyncEstFail

  handles DM_BIG_SYNC_EST_FAIL_IND
- HciLeBigSyncLostEvt_t bigSyncLost

  handles DM_BIG_SYNC_LOST_IND
- HciLeBigTermSyncCmplEvt_t bigSyncStop

  handles DM_BIG_SYNC_STOP_IND
- HciLeBigInfoAdvRptEvt_t bigInfoAdvRpt

  handles DM_BIG_INFO_ADV_REPORT_IND
- dmL2cCmdRejEvt_t l2cCmdRej

  handles DM_L2C_CMD_REJ_IND
- hciHwErrorEvt_t hwError

  handles DM_HW_ERROR_IND
- hciVendorSpecEvt_t vendorSpec

  handles DM_VENDOR_SPEC_IND

2.4.1 Detailed Description

Union of DM callback event data types.

Note

the following events use only the common wsfMsgHdr_t header: DM_RESET_CMPL_IND, DM_ADV_STA↩RT_IND, DM_ADV_STOP_IND, DM_SCAN_START_IND, DM_SCAN_STOP_IND, DM_SEC_PAIR_FAIL_↩IND, DM_SEC_ENCRYPT_FAIL_IND, DM_PRIV_RESOLVED_ADDR_IND, DM_EXT_SCAN_START_IND, DM_EXT_SCAN_STOP_IND, DM_ERROR_IND

Definition at line 807 of file dm_api.h.

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

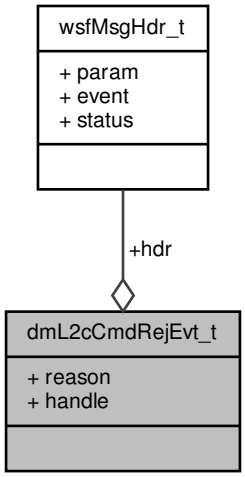
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.5 dmL2cCmdRejEvt_t Struct Reference

Data structure for DM_L2C_CMD_REJ_IND.

`#include <dm_api.h>`

Collaboration diagram for dmL2cCmdRejEvt_t:



Data Fields

- wsfMsgHdr_t hdr

  Header.
- uint16_t reason

  Rejection reason.
- uint16_t handle

  Connection handle.

2.5.1 Detailed Description

Data structure for DM_L2C_CMD_REJ_IND.

Definition at line 785 of file dm_api.h.

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

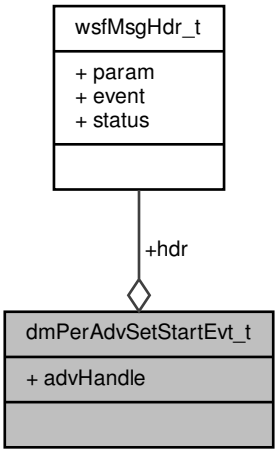
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.6 dmPerAdvSetStartEvt_t Struct Reference

Data structure for DM_PER_ADV_SET_START_IND.

```
#include <dm_api.h>
```

Collaboration diagram for dmPerAdvSetStartEvt_t:



Data Fields

- wsfMsgHdr_t hdr

  Header.
- uint8_t advHandle

  Advertising handle.

2.6.1 Detailed Description

Data structure for DM_PER_ADV_SET_START_IND.

Definition at line 753 of file dm_api.h.

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

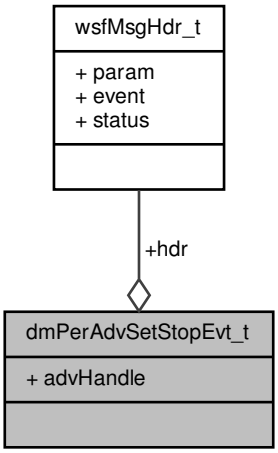
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.7 dmPerAdvSetStopEvt_t Struct Reference

Data structure for DM_PER_ADV_SET_STOP_IND.

`#include <dm_api.h>`

Collaboration diagram for dmPerAdvSetStopEvt_t:



Data Fields

- wsfMsgHdr_t hdr

  Header.
- uint8_t advHandle

  Advertising handle.

2.7.1 Detailed Description

Data structure for DM_PER_ADV_SET_STOP_IND.

Definition at line 760 of file dm_api.h.

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

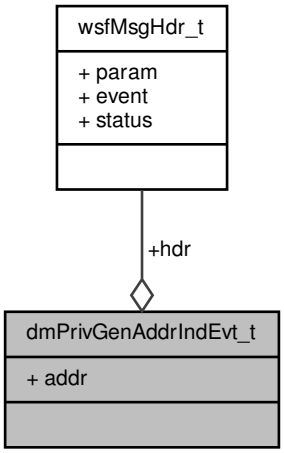
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.8 dmPrivGenAddrIndEvt_t Struct Reference

Data type for DM_PRIV_GENERATE_ADDR_IND.

`#include <dm_api.h>`

Collaboration diagram for dmPrivGenAddrIndEvt_t:



Data Fields

- wsfMsgHdr_t hdr

  Header.
- bdAddr_t addr

  Resolvable private address.

2.8.1 Detailed Description

Data type for DM_PRIV_GENERATE_ADDR_IND.

Definition at line 722 of file dm_api.h.

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

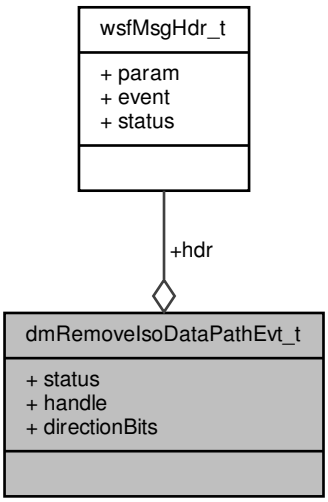
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.9 dmRemoveIsoDataPathEvt_t Struct Reference

Data structure for DM_ISO_DATA_PATH_REMOVE_IND.

`#include <dm_api.h>`

Collaboration diagram for dmRemoveIsoDataPathEvt_t:



Data Fields

- wsfMsgHdr_t hdr
  Event header.
- uint8_t status
  Status.
- uint8_t handle
  Connection handle of the CIS or BIS.
- uint8_t directionBits
  Data path directions being removed.

2.9.1 Detailed Description

Data structure for DM_ISO_DATA_PATH_REMOVE_IND.

Definition at line 776 of file dm_api.h.

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

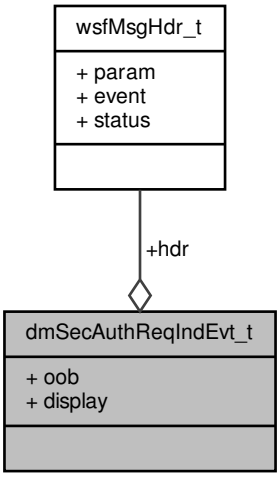
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.10 dmSecAuthReqIndEvt_t Struct Reference

Data type for DM_SEC_AUTH_REQ_IND.

```
#include <dm_api.h>
```

Collaboration diagram for dmSecAuthReqIndEvt_t:



Data Fields

- wsfMsgHdr_t hdr

  Header.
- bool_t oob

  Out-of-band data requested.
- bool_t display

  TRUE if pin is to be displayed.

2.10.1 Detailed Description

Data type for DM_SEC_AUTH_REQ_IND.

Definition at line 673 of file dm_api.h.

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

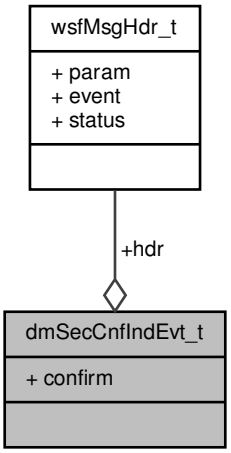
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.11 dmSecCnfIndEvt_t Struct Reference

Data type for DM_SEC_COMPARE_IND.

`#include <dm_api.h>`

Collaboration diagram for dmSecCnfIndEvt_t:



Data Fields

- wsfMsgHdr_t hdr

  Header.
- uint8_t confirm [SMP_CONFIRM_LEN]

  Confirm value.

2.11.1 Detailed Description

Data type for DM_SEC_COMPARE_IND.

Definition at line 708 of file dm_api.h.

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

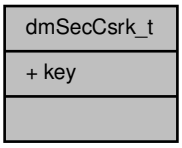
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.12 dmSecCsrk_t Struct Reference

CSRK data type.

```
#include <dm_api.h>
```

Collaboration diagram for dmSecCsrk_t:



Data Fields

- uint8_t key [SMP_KEY_LEN]
  CSRK.

2.12.1 Detailed Description

CSRK data type.

Definition at line 645 of file dm_api.h.

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

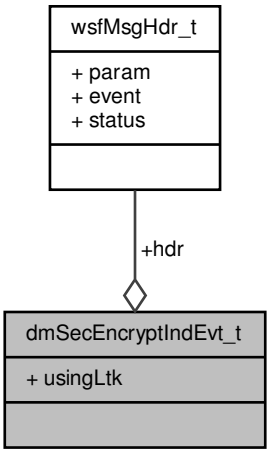
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.13 dmSecEncryptIndEvt_t Struct Reference

Data type for DM_SEC_ENCRYPT_IND.

```
#include <dm_api.h>
```

Collaboration diagram for dmSecEncryptIndEvt_t:



Data Fields

- wsfMsgHdr_t hdr

  Header.
- bool_t usingLtk

  TRUE if connection encrypted with LTK.

2.13.1 Detailed Description

Data type for DM_SEC_ENCRYPT_IND.

Definition at line 666 of file dm_api.h.

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

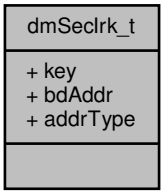
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.14 dmSecIrk_t Struct Reference

IRK data type.

```
#include <dm_api.h>
```

Collaboration diagram for dmSecIrk_t:



Data Fields

- uint8_t key [SMP_KEY_LEN]

  IRK.
- bdAddr_t bdAddr

  BD Address.
- uint8_t addrType

  Address Type.

2.14.1 Detailed Description

IRK data type.

Definition at line 637 of file dm_api.h.

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

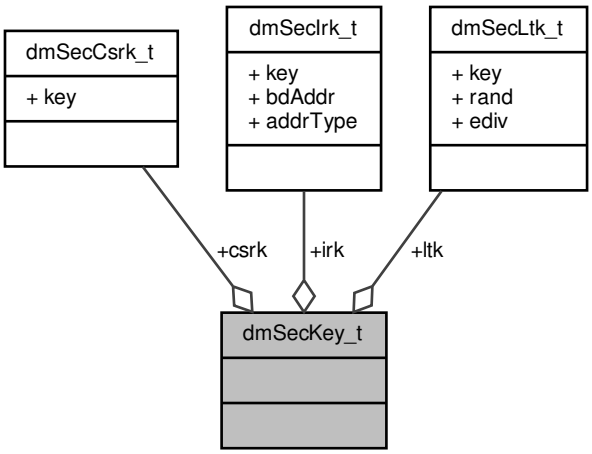
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.15 dmSecKey_t Union Reference

Union of key types.

```
#include <dm_api.h>
```

Collaboration diagram for dmSecKey_t:



Data Fields

- dmSecLtk_t ltk
  LTK.
- dmSecIrk_t irk
  IRK.
- dmSecCsrk_t csrk
  CSRK.

2.15.1 Detailed Description

Union of key types.

Definition at line 651 of file dm_api.h.

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

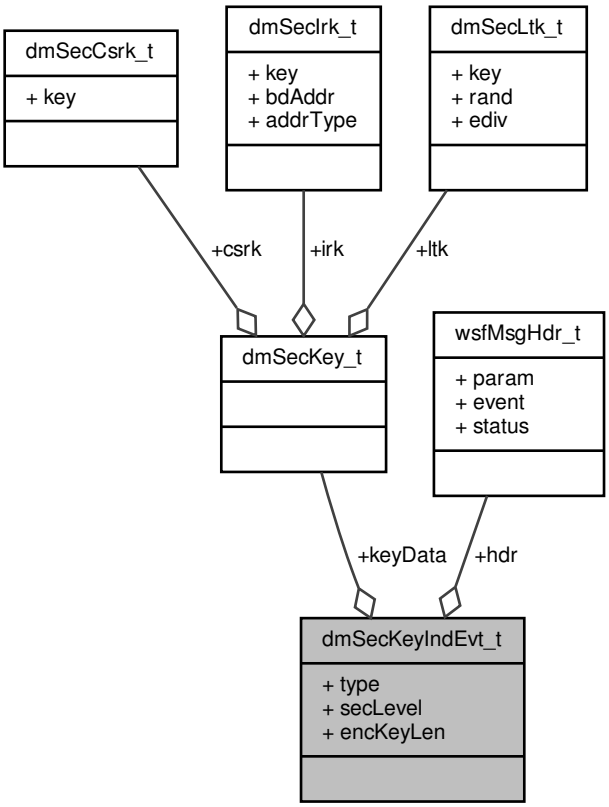
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.16 dmSecKeyIndEvt_t Struct Reference

Data type for DM_SEC_KEY_IND.

```
#include <dm_api.h>
```

Collaboration diagram for dmSecKeyIndEvt_t:



Data Fields

- wsfMsgHdr_t hdr
  Header.
- dmSecKey_t keyData
  Key data.
- uint8_t type
  Key type.
- uint8_t secLevel
  Security level of pairing when key was exchanged.
- uint8_t encKeyLen
  Length of encryption key used when data was transferred.

2.16.1 Detailed Description

Data type for DM_SEC_KEY_IND.

Definition at line 698 of file dm_api.h.

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

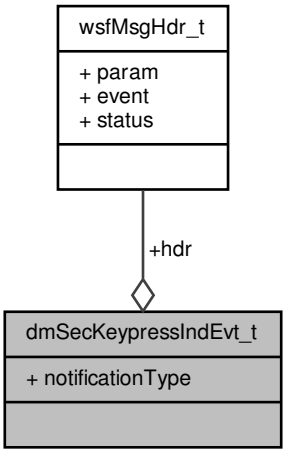
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.17 dmSecKeypressIndEvt_t Struct Reference

Data type for DM_SEC_KEYPRESS_IND.

`#include <dm_api.h>`

Collaboration diagram for dmSecKeypressIndEvt_t:



Data Fields

- wsfMsgHdr_t hdr

  Header.
- uint8_t notificationType

  Type of keypress notification.

2.17.1 Detailed Description

Data type for DM_SEC_KEYPRESS_IND.

Definition at line 715 of file dm_api.h.

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

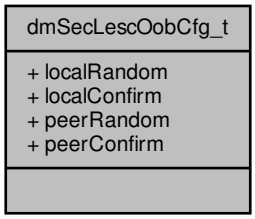
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.18 dmSecLescOobCfg_t Struct Reference

Data type for DmSecSetOob().

```
#include <dm_api.h>
```

Collaboration diagram for dmSecLescOobCfg_t:



Data Fields

- uint8_t localRandom [SMP_RAND_LEN]

  Random value of the local device.
- uint8_t localConfirm [SMP_CONFIRM_LEN]

  Confirm value of the local device.
- uint8_t peerRandom [SMP_RAND_LEN]

  Random value of the peer device.
- uint8_t peerConfirm [SMP_CONFIRM_LEN]

  Confirm value of the peer device.

2.18.1 Detailed Description

Data type for DmSecSetOob().

Definition at line 903 of file dm_api.h.

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

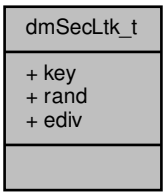
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.19 dmSecLtk_t Struct Reference

LTK data type.

`#include <dm_api.h>`

Collaboration diagram for dmSecLtk_t:



Data Fields

- uint8_t key [SMP_KEY_LEN]

  LTK.
- uint8_t rand [SMP_RAND8_LEN]

  Rand.
- uint16_t ediv

  EDIV.

2.19.1 Detailed Description

LTK data type.

Definition at line 629 of file dm_api.h.

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

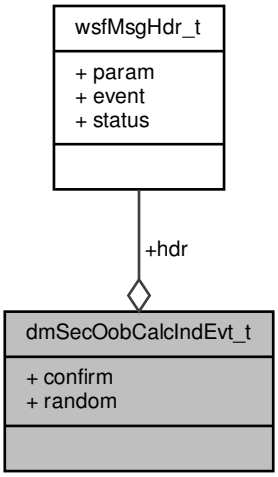
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.20 dmSecOobCalcIndEvt_t Struct Reference

Data type for DM_SEC_CALC_OOB_IND.

`#include <dm_api.h>`

Collaboration diagram for dmSecOobCalcIndEvt_t:



Data Fields

- wsfMsgHdr_t hdr

  Header.
- uint8_t confirm [SMP_CONFIRM_LEN]

  Local confirm value.
- uint8_t random [SMP_RAND_LEN]

  Local random value.

2.20.1 Detailed Description

Data type for DM_SEC_CALC_OOB_IND.

Definition at line 729 of file dm_api.h.

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

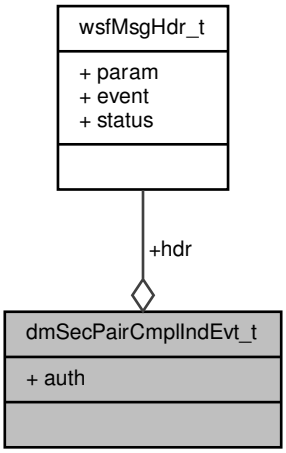
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.21 dmSecPairCmplIndEvt_t Struct Reference

Data type for DM_SEC_PAIR_CMPL_IND.

`#include <dm_api.h>`

Collaboration diagram for dmSecPairCmplIndEvt_t:



Data Fields

- wsfMsgHdr_t hdr

  Header.
- uint8_t auth

  Authentication and bonding flags.

2.21.1 Detailed Description

Data type for DM_SEC_PAIR_CMPL_IND.

Definition at line 659 of file dm_api.h.

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

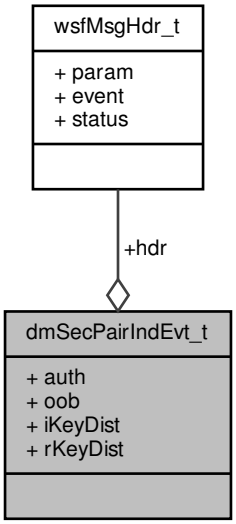
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.22 dmSecPairIndEvt_t Struct Reference

Data type for DM_SEC_PAIR_IND.

`#include <dm_api.h>`

Collaboration diagram for dmSecPairIndEvt_t:



Data Fields

- wsfMsgHdr_t hdr
  Header.
- uint8_t auth
  Authentication and bonding flags.
- bool_t oob
  Out-of-band pairing data present or not present.
- uint8_t iKeyDist
  Initiator key distribution flags.
- uint8_t rKeyDist
  Responder key distribution flags.

2.22.1 Detailed Description

Data type for DM_SEC_PAIR_IND.

Definition at line 681 of file dm_api.h.

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

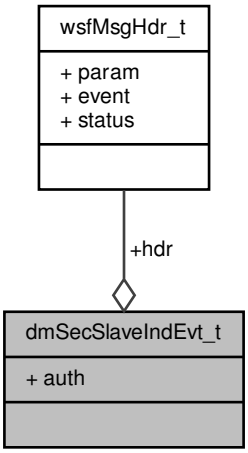
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.23 dmSecSlaveIndEvt_t Struct Reference

Data type for DM_SEC_SLAVE_REQ_IND.

`#include <dm_api.h>`

Collaboration diagram for dmSecSlaveIndEvt_t:



Data Fields

- wsfMsgHdr_t hdr

  Header.
- uint8_t auth

  Authentication and bonding flags.

2.23.1 Detailed Description

Data type for DM_SEC_SLAVE_REQ_IND.

Definition at line 691 of file dm_api.h.

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

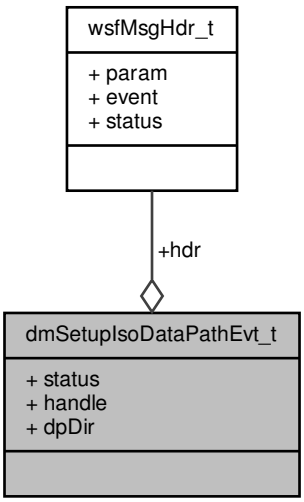
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.24 dmSetupIsoDataPathEvt_t Struct Reference

Data structure for DM_ISO_DATA_PATH_SETUP_IND.

`#include <dm_api.h>`

Collaboration diagram for dmSetupIsoDataPathEvt_t:



Data Fields

- wsfMsgHdr_t hdr
  Event header.
- uint8_t status
  Status.
- uint8_t handle
  Connection handle of the CIS or BIS.
- uint8_t dpDir
  Data path direction being set up.

2.24.1 Detailed Description

Data structure for DM_ISO_DATA_PATH_SETUP_IND.

Definition at line 767 of file dm_api.h.

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

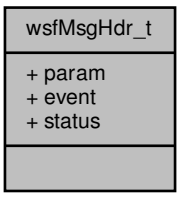
- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h

2.25 wsfMsgHdr_t Struct Reference

Common message structure passed to event handler.

`#include <wsf_os.h>`

Collaboration diagram for wsfMsgHdr_t:



Data Fields

- uint16_t param

  General purpose parameter passed to event handler.
- uint8_t event

  General purpose event value passed to event handler.
- uint8_t status

  General purpose status value passed to event handler.

2.25.1 Detailed Description

Common message structure passed to event handler.

Definition at line 132 of file wsf_os.h.

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

- /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/wsf/include/wsf_os.h

Chapter 3

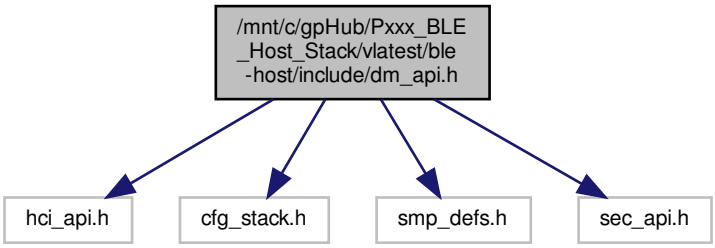
File Documentation

3.1 /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h File Reference

Device Manager subsystem API.

```
#include "hci_api.h"
#include "cfg_stack.h"
#include "smp_defs.h"
#include "sec_api.h"
```

Include dependency graph for dm_api.h:



Data Structures

- struct dmCfg_t

  Configuration structure.
- struct dmSecLtk_t

  LTK data type.
- struct dmSecIrk_t

  IRK data type.
- struct dmSecCsrk_t

CSRK data type.

- union dmSecKey_t

  Union of key types.

- struct dmSecPairCmplIndEvt_t

  Data type for DM_SEC_PAIR_CMPL_IND.

- struct dmSecEncryptIndEvt_t

  Data type for DM_SEC_ENCRYPT_IND.

- struct dmSecAuthReqIndEvt_t

  Data type for DM_SEC_AUTH_REQ_IND.

- struct dmSecPairIndEvt_t

  Data type for DM_SEC_PAIR_IND.

- struct dmSecSlaveIndEvt_t

  Data type for DM_SEC_SLAVE_REQ_IND.

- struct dmSecKeyIndEvt_t

  Data type for DM_SEC_KEY_IND.

- struct dmSecCnfIndEvt_t

  Data type for DM_SEC_COMPARE_IND.

- struct dmSecKeypressIndEvt_t

  Data type for DM_SEC_KEYPRESS_IND.

- struct dmPrivGenAddrIndEvt_t

  Data type for DM_PRIV_GENERATE_ADDR_IND.

- struct dmSecOobCalcIndEvt_t

  Data type for DM_SEC_CALC_OOB_IND.

- struct dmAdvNewAddrIndEvt_t

  Data type for DM_ADV_NEW_ADDR_IND.

- struct dmAdvSetStartEvt_t

  Data structure for DM_ADV_SET_START_IND.

- struct dmPerAdvSetStartEvt_t

  Data structure for DM_PER_ADV_SET_START_IND.

- struct dmPerAdvSetStopEvt_t

  Data structure for DM_PER_ADV_SET_STOP_IND.

- struct dmSetupIsoDataPathEvt_t

  Data structure for DM_ISO_DATA_PATH_SETUP_IND.

- struct dmRemoveIsoDataPathEvt_t

  Data structure for DM_ISO_DATA_PATH_REMOVE_IND.

- struct dmL2cCmdRejEvt_t

  Data structure for DM_L2C_CMD_REJ_IND.

- union dmEvt_t

  Union of DM callback event data types.

- struct dmSecLescOobCfg_t

  Data type for DmSecSetOob().

Macros

- #define DM_SEC_HCI_ERR_BASE 0x20

  Base value for HCI error status values for DM_SEC_PAIR_CMPL_IND.

GAP Device Role

Connectable GAP Roles.

- #define DM_ROLE_MASTER HCI_ROLE_MASTER
  Role is master.
- #define DM_ROLE_SLAVE HCI_ROLE_SLAVE
  Role is slave.

GAP Discovery Mode

When setup as a discoverable device, these are the possible modes of discovery.

- #define DM_DISC_MODE_NONE 0
  GAP non-discoverable.
- #define DM_DISC_MODE_LIMITED 1
  GAP limited discoverable mode.
- #define DM_DISC_MODE_GENERAL 2
  GAP general discoverable mode.

GAP Advertising Type

Type of connectable or disconverable advertising to perform.

- #define DM_ADV_CONN_UNDIRECT 0
  Connectable and scannable undirected advertising.
- #define DM_ADV_CONN_DIRECT 1
  Connectable directed advertising.
- #define DM_ADV_SCAN_UNDIRECT 2
  Scannable undirected advertising.
- #define DM_ADV_NONCONN_UNDIRECT 3
  Non-connectable and non-scannable undirected advertising.
- #define DM_ADV_CONN_DIRECT_LO_DUTY 4
  Connectable directed low duty cycle advertising.

GAP AE Advertising Types

Advertising extension types - AE only.

- #define DM_EXT_ADV_CONN_UNDIRECT 5
  Connectable undirected advertising.
- #define DM_EXT_ADV_NONCONN_DIRECT 6
  Non-connectable and non-scannable directed advertising.
- #define DM_EXT_ADV_SCAN_DIRECT 7
  Scannable directed advertising.
- #define DM_ADV_NONE 255
  For internal use only.

GAP Advertising Report Type

Type of an advertising report observed while scanning.

- #define DM_RPT_CONN_UNDIRECT 0
  Connectable and scannable undirected advertising.
- #define DM_RPT_CONN_DIRECT 1
  Connectable directed advertising.
- #define DM_RPT_SCAN_UNDIRECT 2
  Scannable undirected advertising.
- #define DM_RPT_NONCONN_UNDIRECT 3
  Non-connectable undirected advertising.
- #define DM_RPT_SCAN_RESPONSE 4
  Scan response.

GAP Advertising Data Location

Whether data is located in the advertising data or in the scan response data

- #define DM_DATA_LOC_ADV 0
  Locate data in the advertising data.
- #define DM_DATA_LOC_SCAN 1
  Locate data in the scan response data.

GAP Scan Type

When setup as a connectable or observer device, this is the type of scanning to perform.

- #define DM_SCAN_TYPE_PASSIVE 0
  Passive scan.
- #define DM_SCAN_TYPE_ACTIVE 1
  Active scan.

GAP Advertising Channel Map

Advertising channel map codes

- #define DM_ADV_CHAN_37 HCI_ADV_CHAN_37
  Advertising channel 37.
- #define DM_ADV_CHAN_38 HCI_ADV_CHAN_38
  Advertising channel 38.
- #define DM_ADV_CHAN_39 HCI_ADV_CHAN_39
  Advertising channel 39.
- #define DM_ADV_CHAN_ALL (HCI_ADV_CHAN_37 | HCI_ADV_CHAN_38 | HCI_ADV_CHAN_39)
  All advertising channels.

DM Client IDs

The client ID parameter to function DmConnRegister()

- #define DM_CLIENT_ID_ATT 0
  Identifier for attribute protocol, for internal use only.
- #define DM_CLIENT_ID_SMP 1
  Identifier for security manager protocol, for internal use only.
- #define DM_CLIENT_ID_DM 2
  Identifier for device manager, for internal use only.
- #define DM_CLIENT_ID_APP 3
  Identifier for the application.
- #define DM_CLIENT_ID_L2C 4
  Identifier for L2CAP.
- #define DM_CLIENT_ID_MAX 5
  For internal use only.

DM Unknown IDs

Values for unknown or unspecificed device identifiers.

- #define DM_CONN_ID_NONE 0
  Unknown connection ID or other error.
- #define DM_SYNC_ID_NONE 0
  Unknown sync ID or other error.
- #define DM_CIG_ID_NONE 0xFF
  Unknown Connected Isochronous Group (CIG) ID or other error.
- #define DM_CIS_ID_NONE 0xFF
  Unknown Connected Isochronous Stream (CIS) ID or other error.

GAP Address Type

The address type to use over the air or that is associated with a received address.

- #define DM_ADDR_PUBLIC 0x00
  Public device address.
- #define DM_ADDR_RANDOM 0x01
  Random device address.
- #define DM_ADDR_PUBLIC_IDENTITY 0x02
  Public identity address (corresponds to resolved private address)
- #define DM_ADDR_RANDOM_IDENTITY 0x03
  Random (static) identity address (corresponds to resolved private address)
- #define DM_ADDR_RANDOM_UNRESOLVED 0xFE
  Random device address (Controller unable to resolve)
- #define DM_ADDR_NONE 0xFF
  No address provided (anonymous)

GAP Advertising Data Types

Advertising data types flags.

- #define DM_ADV_TYPE_FLAGS 0x01
  Flag bits.
- #define DM_ADV_TYPE_16_UUID_PART 0x02
  Partial list of 16 bit UUIDs.
- #define DM_ADV_TYPE_16_UUID 0x03
  Complete list of 16 bit UUIDs.
- #define DM_ADV_TYPE_32_UUID_PART 0x04
  Partial list of 32 bit UUIDs.
- #define DM_ADV_TYPE_32_UUID 0x05
  Complete list of 32 bit UUIDs.
- #define DM_ADV_TYPE_128_UUID_PART 0x06
  Partial list of 128 bit UUIDs.
- #define DM_ADV_TYPE_128_UUID 0x07
  Complete list of 128 bit UUIDs.
- #define DM_ADV_TYPE_SHORT_NAME 0x08
  Shortened local name.
- #define DM_ADV_TYPE_LOCAL_NAME 0x09
  Complete local name.
- #define DM_ADV_TYPE_TX_POWER 0x0A
  TX power level.
- #define DM_ADV_TYPE_SM_TK_VALUE 0x10
  Security manager TK value.
- #define DM_ADV_TYPE_SM_OOB_FLAGS 0x11
  Security manager OOB flags.
- #define DM_ADV_TYPE_CONN_INTERVAL 0x12
  Slave preferred connection interval.
- #define DM_ADV_TYPE_SIGNED_DATA 0x13
  Signed data.
- #define DM_ADV_TYPE_16_SOLICIT 0x14
  Service soliticiation list of 16 bit UUIDs.
- #define DM_ADV_TYPE_128_SOLICIT 0x15
  Service soliticiation list of 128 bit UUIDs.
- #define DM_ADV_TYPE_SERVICE_DATA 0x16
  Service data - 16-bit UUID.
- #define DM_ADV_TYPE_PUBLIC_TARGET 0x17
  Public target address.
- #define DM_ADV_TYPE_RANDOM_TARGET 0x18
  Random target address.
- #define DM_ADV_TYPE_APPEARANCE 0x19

Device appearance.

- #define DM_ADV_TYPE_ADV_INTERVAL 0x1A
  Advertising interval.
- #define DM_ADV_TYPE_BD_ADDR 0x1B
  LE Bluetooth device address.
- #define DM_ADV_TYPE_ROLE 0x1C
  LE role.
- #define DM_ADV_TYPE_32_SOLICIT 0x1F
  Service soliticiation list of 32 bit UUIDs.
- #define DM_ADV_TYPE_SVC_DATA_32 0x20
  Service data - 32-bit UUID.
- #define DM_ADV_TYPE_SVC_DATA_128 0x21
  Service data - 128-bit UUID.
- #define DM_ADV_TYPE_LESC_CONFIRM 0x22
  LE Secure Connections confirm value.
- #define DM_ADV_TYPE_LESC_RANDOM 0x23
  LE Secure Connections random value.
- #define DM_ADV_TYPE_URI 0x24
  URI.
- #define DM_ADV_TYPE_INDOOR_POS 0x25
  Indoor positioning service.
- #define DM_ADV_TYPE_TRANS_DISC 0x26
  Transport discovery service.
- #define DM_ADV_TYPE_LE_SUP_FEAT 0x27
  LE supported features.
- #define DM_ADV_TYPE_CH_MAP_UPD_IND 0x28
  Channel map update indication.
- #define DM_ADV_TYPE_PB_ADV 0x29
  PB-ADV.
- #define DM_ADV_TYPE_MESH_MSG 0x2A
  Mesh message.
- #define DM_ADV_TYPE_MESH_BEACON 0x2B
  Mesh beacon.
- #define DM_ADV_TYPE_BIG_INFO 0x2C
  BIG Info.
- #define DM_ADV_TYPE_BCAST_CODE 0x2D
  Mesh beacon.
- #define DM_ADV_TYPE_3D_INFO_DATA 0x3D
  3D information data
- #define DM_ADV_TYPE_MANUFACTURER 0xFF
  Manufacturer specific data.

GAP Advertising Data Flag Advertising Type

Bit mask for Advertising Type flag in advertising data.

- #define DM_FLAG_LE_LIMITED_DISC 0x01
  Limited discoverable flag.
- #define DM_FLAG_LE_GENERAL_DISC 0x02
  General discoverable flag.
- #define DM_FLAG_LE_BREDR_NOT_SUP 0x04
  BR/EDR not supported flag.

GAP Advertising Data Element Indexes

Advertising data element indexes.

- #define DM_AD_LEN_IDX 0
  Advertising data element len.

- #define DM_AD_TYPE_IDX 1
  Advertising data element type.
- #define DM_AD_DATA_IDX 2
  Advertising data element data.

GAP Advertising URI

Advertising URI Scheme

- #define DM_URI_SCHEME_HTTP 0x16
  URI HTTP Scheme.
- #define DM_URI_SCHEME_HTTPS 0x17
  URI HTTPS Scheme.

GAP Timeouts

Timeouts defined by the GAP specification; in units of milliseconds.

- #define DM_GAP_LIM_ADV_TIMEOUT 180000
  Maximum advertising duration in limited discoverable mode.
- #define DM_GAP_GEN_DISC_SCAN_MIN 10240
  Minimum scan duration for general discovery.
- #define DM_GAP_LIM_DISC_SCAN_MIN 10240
  Minimum scan duration for limited discovery.
- #define DM_GAP_CONN_PARAM_TIMEOUT 30000
  Connection parameter update timeout.
- #define DM_GAP_SCAN_FAST_PERIOD 30720
  Minimum time to perform scanning when user initiated.
- #define DM_GAP_ADV_FAST_PERIOD 30000
  Minimum time to perform advertising when user initiated.

GAP 1M PHY Timing

Advertising, scanning, and connection parameters defined in the GAP specification for the LE 1M PHY. In units of 625 microseconds.

- #define DM_GAP_SCAN_FAST_INT_MIN 48
  Minimum scan interval when user initiated.
- #define DM_GAP_SCAN_FAST_INT_MAX 96
  Maximum scan interval when user initiated.
- #define DM_GAP_SCAN_FAST_WINDOW 48
  Scan window when user initiated.
- #define DM_GAP_SCAN_SLOW_INT_1 2048
  Scan interval 1 when background scannning.
- #define DM_GAP_SCAN_SLOW_WINDOW_1 18
  Scan window 1 when background scanning.
- #define DM_GAP_SCAN_SLOW_INT_2 4096
  Scan interval 2 when background scannning.
- #define DM_GAP_SCAN_SLOW_WINDOW_2 36
  Scan window 2 when background scanning.
- #define DM_GAP_ADV_FAST_INT_MIN_1 48
  Minimum advertising interval 1 when user initiated.
- #define DM_GAP_ADV_FAST_INT_MAX_1 96
  Maximum advertising interval 1 when user initiated.
- #define DM_GAP_ADV_FAST_INT_MIN_2 160
  Minimum advertising interval 2 when user initiated.
- #define DM_GAP_ADV_FAST_INT_MAX_2 240
  Maximum advertising interval 2 when user initiated.
- #define DM_GAP_ADV_SLOW_INT_MIN 1600

Minimum advertising interval when background advertising.

- #define DM_GAP_ADV_SLOW_INT_MAX 1920

  Maximum advertising interval when background advertising.

GAP Coded PHY Timing

Advertising, scanning, and connection parameters defined in the GAP specification for the LE Coded PHY. In units of 625 microseconds.

- #define DM_GAP_SCAN_CODED_FAST_INT_MIN 144

  Minimum scan interval when user initiated on LE Coded PHY.
- #define DM_GAP_SCAN_CODED_FAST_INT_MAX 288

  Maximum scan interval when user initiated on LE Coded PHY.
- #define DM_GAP_SCAN_CODED_FAST_WINDOW 144

  Scan window when user initiated on LE Coded PHY.
- #define DM_GAP_SCAN_CODED_SLOW_INT_1 6144

  Scan interval 1 when background scannning on LE Coded PHY.
- #define DM_GAP_SCAN_CODED_SLOW_WINDOW_1 54

  Scan window 1 when background scanning on LE Coded PHY.
- #define DM_GAP_SCAN_CODED_SLOW_INT_2 12288

  Scan interval 2 when background scannning on LE Coded PHY.
- #define DM_GAP_SCAN_CODED_SLOW_WINDOW_2 108

  Scan window 2 when background scanning on LE Coded PHY.
- #define DM_GAP_ADV_CODED_FAST_INT_MIN_1 144

  Minimum advertising interval 1 when user initiated on LE Coded PHY.
- #define DM_GAP_ADV_CODED_FAST_INT_MAX_1 288

  Maximum advertising interval 1 when user initiated on LE Coded PHY.
- #define DM_GAP_ADV_CODED_FAST_INT_MIN_2 480

  Minimum advertising interval 2 when user initiated on LE Coded PHY.
- #define DM_GAP_ADV_CODED_FAST_INT_MAX_2 720

  Maximum advertising interval 2 when user initiated on LE Coded PHY.
- #define DM_GAP_ADV_CODED_SLOW_INT_MIN 4800

  Minimum advertising interval when background advertising on LE Coded PHY.
- #define DM_GAP_ADV_CODED_SLOW_INT_MAX 5760

  Maximum advertising interval when background advertising on LE Coded PHY.

GAP Connection Slave Latency

- #define DM_GAP_CONN_EST_LATENCY 0

  GAP connection establishment slaves latency.

GAP Connection Interval

GAP connection interval in 1.25ms units.

- #define DM_GAP_INITIAL_CONN_INT_MIN 24

  Minimum initial connection interval.
- #define DM_GAP_INITIAL_CONN_INT_MAX 40

  Maximum initial connection interval.

GAP Connection Event Lengths

GAP connection establishment minimum and maximum connection event lengths.

- #define DM_GAP_CONN_EST_MIN_CE_LEN 0

  Connection establishment minimum event length.
- #define DM_GAP_CONN_EST_MAX_CE_LEN 0

  Connection establishment maximum event length.

GAP Peripheral Privacy Characteristic Values

- #define DM_GAP_PRIV_DISABLED 0
  Privacy Disabled.
- #define DM_GAP_PRIV_ENABLED 1
  Privacy Enabled.

GAP Connection Supervision Timeout

Connection supervision timeout, in 10ms units

- #define DM_DEFAULT_EST_SUP_TIMEOUT 2000
  Connection establishment supervision timeout default, in 10ms units.

GAP Security Pairing Authentication Requirements

Pairing authentication/security properties bit mask.

- #define DM_AUTH_BOND_FLAG SMP_AUTH_BOND_FLAG
  Bonding requested.
- #define DM_AUTH_MITM_FLAG SMP_AUTH_MITM_FLAG
  MITM (authenticated pairing) requested.
- #define DM_AUTH_SC_FLAG SMP_AUTH_SC_FLAG
  LE Secure Connections requested.
- #define DM_AUTH_KP_FLAG SMP_AUTH_KP_FLAG
  Keypress notifications requested.

GAP Key Distribution Flags

Key distribution bit mask

- #define DM_KEY_DIST_LTK SMP_KEY_DIST_ENC
  Distribute LTK used for encryption.
- #define DM_KEY_DIST_IRK SMP_KEY_DIST_ID
  Distribute IRK used for privacy.
- #define DM_KEY_DIST_CSRK SMP_KEY_DIST_SIGN
  Distribute CSRK used for signed data.

DM Security Key Indication Types

Type of key used in DM_SEC_KEY_IND.

- #define DM_KEY_LOCAL_LTK 0x01
  LTK generated locally for this device.
- #define DM_KEY_PEER_LTK 0x02
  LTK received from peer device.
- #define DM_KEY_IRK 0x04
  IRK and identity info of peer device.
- #define DM_KEY_CSRK 0x08
  CSRK of peer device.

GAP Security Level

GAP Mode 1 Security Levels

- #define DM_SEC_LEVEL_NONE 0
  Connection has no security.
- #define DM_SEC_LEVEL_ENC 1
  Connection is encrypted with unauthenticated key.
- #define DM_SEC_LEVEL_ENC_AUTH 2

Connection is encrypted with authenticated key.

- #define DM_SEC_LEVEL_ENC_LESC 3

  Connection is encrypted with LE Secure Connections.

GAP Broadcast Security Level

GAP Mode 3 Security Levels

- #define DM_SEC_LEVEL_BCAST_NONE 0

  No security (no authentication and no encryption)
- #define DM_SEC_LEVEL_BCAST_UNAUTH 1

  Use of unauthenticated Broadcast_Code.
- #define DM_SEC_LEVEL_BCAST_AUTH 2

  Use of authenticated Broadcast_Code.

GAP Random Address Types

Random address type masks.

- #define DM_RAND_ADDR_STATIC 0xC0

  Static address.
- #define DM_RAND_ADDR_RESOLV 0x40

  Resolvable private address.
- #define DM_RAND_ADDR_NONRESOLV 0x00

  Non-resolvable private address.

GAP Random Address Macros

Macros for identifying address type.

- #define DM_RAND_ADDR_GET(addr) ((addr)[5] & 0xC0)

  Get the type of random address.
- #define DM_RAND_ADDR_SET(addr, type) {(addr)[5] = ((addr)[5] & 0x3F) | (type);}

  Set the type of random address.
- #define DM_RAND_ADDR_SA(addr, type)

  Check for Static Address.
- #define DM_RAND_ADDR_RPA(addr, type)

  Check for Resolvable Private Address.

GAP Privacy Mode

Privacy Mode of this device in regards to a peer device.

- #define DM_PRIV_MODE_NETWORK 0x00

  Network privacy mode (default).
- #define DM_PRIV_MODE_DEVICE 0x01

  Device privacy mode.

DM Internal State

Connection busy or idle state

- #define DM_CONN_IDLE 0

  Connection is idle.
- #define DM_CONN_BUSY 1

  Connection is busy.

DM Internal State Flags

Connection busy/idle state bitmask.

- #define DM_IDLE_SMP_PAIR 0x0001
  SMP pairing in progress.
- #define DM_IDLE_DM_ENC 0x0002
  DM Encryption setup in progress.
- #define DM_IDLE_ATTS_DISC 0x0004
  ATTS service discovery in progress.
- #define DM_IDLE_APP_DISC 0x0008
  App framework service discovery in progress.
- #define DM_IDLE_USER_1 0x0010
  For use by user application.
- #define DM_IDLE_USER_2 0x0020
  For use by user application.
- #define DM_IDLE_USER_3 0x0040
  For use by user application.
- #define DM_IDLE_USER_4 0x0080
  For use by user application.

GAP Filter Policy Modes

Filter policy modes.

- #define DM_FILT_POLICY_MODE_ADV 0
  Advertising filter policy mode.
- #define DM_FILT_POLICY_MODE_SCAN 1
  Scanning filter policy mode.
- #define DM_FILT_POLICY_MODE_INIT 2
  Initiator filter policy mode.
- #define DM_FILT_POLICY_MODE_SYNC 3
  Synchronization filter policy mode.

DM Proprietary Error Codes

Internal error codes not sent in any PDU.

- #define DM_ERR_SMP_RX_PDU_LEN_EXCEEDED 0x01
  LESC key length exceeded maximum RX PDU length.
- #define DM_ERR_ATT_RX_PDU_LEN_EXCEEDED 0x02
  Configured ATT MTU exceeded maximum RX PDU length.
- #define DM_ERR_L2C_RX_PDU_LEN_EXCEEDED 0x03
  Registered COC MPS exceeded maximum RX PDU length.

DM Legacy Advertising Handle

Default handle for legacy advertising when using legacy HCI interface. In this case only one advertising set is allowed so all activity uses the same handle.

- #define DM_ADV_HANDLE_DEFAULT 0
  Default Advertising handle for legacy advertising.

DM ISO data path directions

Number of ISO data path directions

- #define **DM_ISO_NUM_DIR** 2

Typedefs

- typedef uint8_t dmConnId_t

  Connection identifier.
- typedef uint8_t dmSyncId_t

  Synchronization identifier.
- typedef void(∗ dmCback_t) (dmEvt_t ∗pDmEvt)

  Callback type.

Enumerations

DM Conn CTE states

Internal states of the DM conn CTE.

- enum {
  DM_CONN_CTE_STATE_IDLE,
  DM_CONN_CTE_STATE_INITIATING,
  DM_CONN_CTE_STATE_RESPONDING,
  DM_CONN_CTE_STATE_SAMPLING,
  DM_CONN_CTE_STATE_STARTING,
  DM_CONN_CTE_STATE_STOPPING }

Functions

DM App Callback Registration

- void DmRegister (dmCback_t cback)

  Register a callback with DM for scan and advertising events.

DM Advertising Functions

Functions used to control Legacy and Extended Advertising.

- uint8_t ∗ DmFindAdType (uint8_t adType, uint16_t dataLen, uint8_t ∗pData)

  Find an advertising data element in the given advertising or scan response data.
- void DmAdvInit (void)

  Initialize DM legacy advertising.
- void DmExtAdvInit (void)

  Initialize DM extended advertising.
- bool_t DmAdvModeLeg (void)

  Whether DM advertising is in legacy mode.
- bool_t DmAdvModeExt (void)

  Whether DM advertising is in extended mode.
- void DmAdvConfig (uint8_t advHandle, uint8_t advType, uint8_t peerAddrType, uint8_t ∗pPeerAddr)

  Set the advertising parameters using the given advertising type, and peer address.
- void DmAdvSetData (uint8_t advHandle, uint8_t op, uint8_t location, uint8_t len, uint8_t ∗pData)

  Set the advertising or scan response data to the given data.
- void DmAdvStart (uint8_t numSets, uint8_t ∗pAdvHandles, uint16_t ∗pDuration, uint8_t ∗pMaxEaEvents)

  Start advertising using the given advertising set and duration.
- void DmAdvStop (uint8_t numSets, uint8_t ∗pAdvHandles)

  Stop advertising for the given advertising set. If the number of sets is set to 0 then all advertising sets are disabled.
- void DmAdvRemoveAdvSet (uint8_t advHandle)

  Remove an advertising set.
- void DmAdvClearAdvSets (void)

  Clear advertising sets.
- void DmAdvSetRandAddr (uint8_t advHandle, const uint8_t ∗pAddr)

Set the random device address for a given advertising set.

- void DmAdvSetInterval (uint8_t advHandle, uint16_t intervalMin, uint16_t intervalMax)

  Set the minimum and maximum advertising intervals.
- void DmAdvSetChannelMap (uint8_t advHandle, uint8_t channelMap)

  Include or exclude certain channels from the advertising channel map.
- void DmAdvSetAddrType (uint8_t addrType)

  Set the local address type used while advertising. This function can be used to configure advertising to use a random address.
- bool_t DmAdvSetAdValue (uint8_t adType, uint8_t len, uint8_t *pValue, uint16_t *pAdvDataLen, uint8_t *pAdvData, uint16_t advDataBufLen)

  Set the value of an advertising data element in the given advertising or scan response data. If the element already exists in the data then it is replaced with the new value. If the element does not exist in the data it is appended to it, space permitting.
- bool_t DmAdvSetName (uint8_t len, uint8_t *pValue, uint16_t *pAdvDataLen, uint8_t *pAdvData, uint16_t advDataBufLen)

  Set the device name in the given advertising or scan response data. If the name can only fit in the data if it is shortened, the name is shortened and the AD type is changed to DM_ADV_TYPE_SHORT_NAME.
- void DmDevPrivInit (void)

  Initialize device privacy module.
- void DmDevPrivStart (uint16_t changeInterval)

  Start using a private resolvable address.
- void DmDevPrivStop (void)

  Stop using a private resolvable address.
- void DmAdvUseLegacyPdu (uint8_t advHandle, bool_t useLegacyPdu)

  Set whether or not to use legacy advertising PDUs with extended advertising.
- void DmAdvOmitAdvAddr (uint8_t advHandle, bool_t omitAdvAddr)

  Set whether or not to omit advertiser's address from all PDUs (anonymous advertising).
- void DmAdvIncTxPwr (uint8_t advHandle, bool_t incTxPwr, int8_t advTxPwr)

  Set whether or not to include TxPower in extended header of advertising PDU.
- void DmAdvSetPhyParam (uint8_t advHandle, uint8_t priAdvPhy, uint8_t secAdvMaxSkip, uint8_t secAdvPhy)

  Set extended advertising PHY parameters.
- void DmAdvScanReqNotifEnable (uint8_t advHandle, bool_t scanReqNotifEna)

  Set scan request notification enable.
- void DmAdvSetFragPref (uint8_t advHandle, uint8_t fragPref)

  Set fragment preference for advertising data.
- void DmAdvSetSid (uint8_t advHandle, uint8_t advSid)

  Set advertising SID for the given advertising handle.
- void DmPerAdvConfig (uint8_t advHandle)

  Set the advertising parameters for periodic advertising.
- void DmPerAdvSetData (uint8_t advHandle, uint8_t op, uint8_t len, uint8_t *pData)

  Set the advertising data to the given data for periodic advertising.
- void DmPerAdvStart (uint8_t advHandle)

  Start periodic advertising for the advertising set specified by the advertising handle.
- void DmPerAdvStop (uint8_t advHandle)

  Stop periodic advertising for the advertising set specified by the advertising handle.
- void DmPerAdvSetInterval (uint8_t advHandle, uint16_t intervalMin, uint16_t intervalMax)

  Set the minimum and maximum advertising intervals for periodic advertising.
- void DmPerAdvIncTxPwr (uint8_t advHandle, bool_t incTxPwr)

  Set whether or not to include TxPower in extended header of advertising PDU for periodic advertising.
- bool_t DmPerAdvEnabled (uint8_t advHandle)

  Get status of periodic advertising handle.
- uint16_t DmExtMaxAdvDataLen (uint8_t advType, bool_t useLegacyPdu)

  Get the maximum advertising data length supported by Controller for a given advertising type.

DM Privacy Functions

Functions for controlling Privacy.

- void DmPrivInit (void)

Initialize DM privacy module.

- void DmPrivResolveAddr (uint8_t *pAddr, uint8_t *pIrk, uint16_t param)

  Resolve a private resolvable address. When complete the client's callback function is called with a DM_PRIV_RESOLVED_ADDR_IND event. The client must wait to receive this event before executing this function again.
- void DmPrivAddDevToResList (uint8_t addrType, const uint8_t *pIdentityAddr, uint8_t *pPeerIrk, uint8_t *pLocalIrk, bool_t enableLlPriv, uint16_t param)

  Add device to resolving list. When complete the client's callback function is called with a DM_PRIV_ADD_DEV_TO_RES_LIST_IND event. The client must wait to receive this event before executing this function again.
- void DmPrivRemDevFromResList (uint8_t addrType, const uint8_t *pIdentityAddr, uint16_t param)

  Remove device from resolving list. When complete the client's callback function is called with a DM_PRIV_REM_DEV_FROM_RES_LIST_IND event. The client must wait to receive this event before executing this function again.
- void DmPrivClearResList (void)

  Clear resolving list. When complete the client's callback function is called with a DM_PRIV_CLEAR_RES_LIST_IND event. The client must wait to receive this event before executing this function again.
- void DmPrivReadPeerResolvableAddr (uint8_t addrType, const uint8_t *pIdentityAddr)

  HCI read peer resolvable address command. When complete the client's callback function is called with a DM_PRIV_READ_PEER_RES_ADDR_IND event. The client must wait to receive this event before executing this function again.
- void DmPrivReadLocalResolvableAddr (uint8_t addrType, const uint8_t *pIdentityAddr)

  Read local resolvable address command. When complete the client's callback function is called with a DM_PRIV_READ_LOCAL_RES_ADDR_IND event. The client must wait to receive this event before executing this function again.
- void DmPrivSetAddrResEnable (bool_t enable)

  Enable or disable address resolution in LL. When complete the client's callback function is called with a DM_PRIV_SET_ADDR_RES_ENABLE_IND event. The client must wait to receive this event before executing this function again.
- void DmPrivSetResolvablePrivateAddrTimeout (uint16_t rpaTimeout)

  Set resolvable private address timeout command.
- void DmPrivSetPrivacyMode (uint8_t addrType, const uint8_t *pIdentityAddr, uint8_t mode)

  Set privacy mode for a given entry in the resolving list.
- void DmPrivGenerateAddr (uint8_t *pIrk, uint16_t param)

  Generate a Resolvable Private Address (RPA).
- bool_t DmLlPrivEnabled (void)

  Whether LL Privacy is enabled.

DM Scanner Functions

Functions for controlling Legacy and Extended Scanner behavior.

- void DmScanInit (void)

  Initialize DM legacy scanning.
- void DmExtScanInit (void)

  Initialize DM extended scanning.
- void DmPastInit (void)

  Initialize DM Periodic Advertising Sync Transfer (PAST) module.
- void DmConnCteInit (void)

  Initialize DM Connection Constant Tone Extension (CTE) module.
- bool_t DmScanModeLeg (void)

  Whether DM scanning is in legacy mode.
- bool_t DmScanModeExt (void)

  Whether DM scanning is in extended mode.
- void DmScanStart (uint8_t scanPhys, uint8_t mode, const uint8_t *pScanType, bool_t filterDup, uint16_t duration, uint16_t period)

  Start scanning on the given PHYs.
- void DmScanStop (void)

  Stop scanning.
- void DmScanSetInterval (uint8_t scanPhys, uint16_t *pScanInterval, uint16_t *pScanWindow)

  Set the scan interval and window for the specified PHYs.
- void DmScanSetAddrType (uint8_t addrType)

Set the local address type used while scanning. This function can be used to configure scanning to use a random address.

- dmSyncId_t DmSyncStart (uint8_t advSid, uint8_t advAddrType, const uint8_t *pAdvAddr, uint16_t skip, uint16_t syncTimeout)

  Synchronize with periodic advertising from the given advertiser, and start receiving periodic advertising packets.
- void DmSyncStop (dmSyncId_t syncId)

  Stop reception of the periodic advertising identified by the given sync identifier.
- void DmSyncSetEncrypt (uint16_t syncHandle, bool_t encrypt)

  Set the encryption mode of the Broadcast Isochronous Group (BIG) corresponding to the periodic advertising train identified by the sync handle.
- bool_t DmSyncEncrypted (uint16_t syncHandle)

  Get the encryption mode of the Broadcast Isochronous Group (BIG) corresponding to the periodic advertising train identified by the sync handle.
- bool_t DmSyncEnabled (uint16_t syncHandle)

  Get status of sync identified by the handle.
- void DmSyncInitialRptEnable (bool_t enable)

  DM enable or disable initial periodic advertisement reporting.
- void DmBigSyncStart (uint8_t bigHandle, uint16_t syncHandle, uint8_t mse, uint16_t bigSyncTimeout, uint8_t numBis, uint8_t *pBis)

  Synchronize to a Broadcast Isochronous Group (BIG) described in the periodic advertising train specified by the sync handle.
- void DmBigSyncStop (uint8_t bigHandle)

  Stop synchronizing or cancel the process of synchronizing to the Broadcast Isochronous Group (BIG) identified by the handle.
- bool_t DmBisSyncInUse (uint16_t handle)

  For internal use only. Return TRUE if the BIS sync is in use.
- void DmBigSyncSetBcastCode (uint8_t bigHandle, bool_t encrypt, bool_t authen, uint8_t *pBcastCode)

  Set the Broadcast Code for the given Broadcast Isochronous Group (BIG).
- void DmBigSyncSetSecLevel (uint8_t bigHandle, uint8_t secLevel)

  Set the security level of the LE Security Mode 3 for the given Broadcast Isochronous Group (BIG).
- uint8_t DmBigSyncGetSecLevel (uint16_t handle)

  Get the security level of the LE Security Mode 3 for the given Broadcast Isochronous Group (BIG) connection handle.
- void DmBisMasterInit (void)

  Initialize DM BIS manager for operation as master.
- void DmAddDeviceToPerAdvList (uint8_t advAddrType, uint8_t *pAdvAddr, uint8_t advSid)

  Add device to periodic advertiser list.
- void DmRemoveDeviceFromPerAdvList (uint8_t advAddrType, uint8_t *pAdvAddr, uint8_t advSid)

  DM remove device from periodic advertiser list.
- void DmClearPerAdvList (void)

  DM clear periodic advertiser list.
- void DmPastRptRcvEnable (dmSyncId_t syncId, bool_t enable)

  Enable or disable reports for the periodic advertising identified by the sync id.
- void DmPastSyncTrsf (dmConnId_t connId, uint16_t serviceData, dmSyncId_t syncId)

  Send synchronization information about the periodic advertising identified by the sync id to a connected device.
- void DmPastSetInfoTrsf (dmConnId_t connId, uint16_t serviceData, uint8_t advHandle)

  Send synchronization information about the periodic advertising in an advertising set to a connected device.
- void DmPastConfig (dmConnId_t connId, uint8_t mode, uint16_t skip, uint16_t syncTimeout, uint8_t cte↩ Type)

  Specify how the Controller should process periodic advertising synchronization information received from the device identified by the connnection handle.
- void DmPastDefaultConfig (uint8_t mode, uint16_t skip, uint16_t syncTimeout, uint8_t cteType)

  Specify the initial value for the mode, skip, timeout, and Constant Tone Extension type to be used for all subsequent connections over the LE transport.
- void DmConnCteRxSampleStart (dmConnId_t connId, uint8_t slotDurations, uint8_t switchPatternLen, uint8_t *pAntennaIDs)

  Enable sampling received CTE fields on the specified connection, and configure the antenna switching pattern, and switching and sampling slot durations to be used.
- void DmConnCteRxSampleStop (dmConnId_t connId)

  Disable sampling received CTE fields on the specified connection.

- void DmConnCteTxConfig (dmConnId_t connId, uint8_t cteTypeBits, uint8_t switchPatternLen, uint8_t ∗pAntennaIDs)

  Configure the antenna switching pattern, and permitted CTE types used for transmitting CTEs requested by the peer device on the specified connection.
- void DmConnCteReqStart (dmConnId_t connId, uint16_t cteReqInt, uint8_t reqCteLen, uint8_t reqCteType)

  Initiate the CTE Request procedure on the specified connection.
- void DmConnCteReqStop (dmConnId_t connId)

  Stop initiating the CTE Request procedure on the specified connection.
- void DmConnCteRspStart (dmConnId_t connId)

  Start responding to LL_CTE_REQ PDUs with LL_CTE_RSP PDUs on the specified connection.
- void DmConnCteRspStop (dmConnId_t connId)

  Stop responding to LL_CTE_REQ PDUs with LL_CTE_RSP PDUs on the specified connection.
- uint8_t DmConnCteGetReqState (dmConnId_t connId)

  Returns the device manager's CTE request state for a given connection.
- uint8_t DmConnCteGetRspState (dmConnId_t connId)

  Returns the device manager's CTE response state for a given connection.
- void DmReadAntennaInfo (void)

  Read the switching rates, the sampling rates, the number of antennae, and the maximum length of a transmitted Constant Tone Extension supported by the Controller.

DM Connection Functions

Functions for forming connections and managing connection behavior and parameter updates.

- void DmConnInit (void)

  Initialize DM connection manager.
- void DmConnMasterInit (void)

  Initialize DM connection manager for operation as legacy master.
- void DmExtConnMasterInit (void)

  Initialize DM connection manager for operation as extended master.
- void DmConnSlaveInit (void)

  Initialize DM connection manager for operation as legacy slave.
- void DmExtConnSlaveInit (void)

  Initialize DM connection manager for operation as extended slave.
- void DmConnRegister (uint8_t clientId, dmCback_t cback)

  Register with the DM connection manager.
- dmConnId_t DmConnOpen (uint8_t clientId, uint8_t initPhys, uint8_t addrType, uint8_t ∗pAddr)

  Open a connection to a peer device with the given address.
- void DmConnCancelOpen (void)

  Abort connection open operation.
- void DmConnClose (uint8_t clientId, dmConnId_t connId, uint8_t reason)

  Close the connection with the give connection identifier.
- dmConnId_t DmConnAccept (uint8_t clientId, uint8_t advHandle, uint8_t advType, uint16_t duration, uint8_t maxEaEvents, uint8_t addrType, uint8_t ∗pAddr)

  Accept a connection from the given peer device by initiating directed advertising.
- void DmConnUpdate (dmConnId_t connId, hciConnSpec_t ∗pConnSpec)

  Update the connection parameters of an open connection.
- void DmConnSetScanInterval (uint16_t scanInterval, uint16_t scanWindow)

  Set the scan interval and window for connections to be created with DmConnOpen().
- void DmExtConnSetScanInterval (uint8_t initPhys, uint16_t ∗pScanInterval, uint16_t ∗pScanWindow)

  Set the scan interval and window for extended connections to be created with DmConnOpen().
- void DmConnSetConnSpec (hciConnSpec_t ∗pConnSpec)

  Set the connection spec parameters for connections to be created with DmConnOpen().
- void DmExtConnSetConnSpec (uint8_t initPhys, hciConnSpec_t ∗pConnSpec)

  Set the extended connection spec parameters for extended connections to be created with DmConnOpen().
- void DmConnSetAddrType (uint8_t addrType)

  Set the local address type used for connections created with DmConnOpen().
- void DmConnSetIdle (dmConnId_t connId, uint16_t idleMask, uint8_t idle)

Configure a bit in the connection idle state mask as busy or idle.

- uint16_t DmConnCheckIdle (dmConnId_t connId)

  Check if a connection is idle.
- void DmConnReadRssi (dmConnId_t connId)

  Read RSSI of a given connection.
- void DmRemoteConnParamReqReply (dmConnId_t connId, hciConnSpec_t *pConnSpec)

  Reply to the HCI remote connection parameter request event. This command is used to indicate that the Host has accepted the remote device's request to change connection parameters.
- void DmRemoteConnParamReqNegReply (dmConnId_t connId, uint8_t reason)

  Negative reply to the HCI remote connection parameter request event. This command is used to indicate that the Host has rejected the remote device's request to change connection parameters.
- void DmConnSetDataLen (dmConnId_t connId, uint16_t txOctets, uint16_t txTime)

  Set data length for a given connection.
- uint8_t DmConnRole (dmConnId_t connId)

  Return the connection role indicating master or slave.
- void DmWriteAuthPayloadTimeout (dmConnId_t connId, uint16_t timeout)

  Set authenticated payload timeout for a given connection.
- void DmConnRequestPeerSca (dmConnId_t connId)

  Request the Sleep Clock Accuracy (SCA) of a peer device.

DM CIS Functions

Functions for forming and managing Connected Isochronous Stream (CIS) streams.

- void DmCisInit (void)

  Initialize DM Connected Isochronous Stream (CIS) manager.
- void DmCisMasterInit (void)

  Initialize DM Connected Isochronous Stream (CIS) manager for operation as master.
- void DmCisSlaveInit (void)

  Initialize DM Connected Isochronous Stream (CIS) manager for operation as slave.
- void DmCisCigSetSduInterval (uint8_t cigId, uint32_t sduIntervalMToS, uint32_t sduIntervalSToM)

  Set the interval, in microseconds, of periodic SDUs for the given Connected Isochronous Group (CIG).
- void DmCisCigSetSca (uint8_t cigId, uint8_t sca)

  Set the slaves clock accuracy for the given Connected Isochronous Group (CIG).
- void DmCisCigSetPackingFraming (uint8_t cigId, uint8_t packing, uint8_t framing)

  Set the packing scheme and framing format for the given Connected Isochronous Group (CIG).
- void DmCisCigSetTransLatInterval (uint8_t cigId, uint16_t transLatMToS, uint16_t transLatSToM)

  Set the maximum transport latency, in microseconds, for the given Connected Isochronous Group (CIG).
- void DmCisCigConfig (uint8_t cigId, dmConnId_t numCis, HciCisCisParams_t *pCisParam)

  Set the parameters of one or more Connected Isochronous Streams (CISes) that are associated with the given Connected Isochronous Group (CIG).
- void DmCisCigRemove (uint8_t cigId)

  Remove all the Connected Isochronous Streams (CISes) associated with the given Connected Isochronous Group (CIG).
- void DmCisOpen (uint8_t numCis, uint16_t *pCisHandle, dmConnId_t *pConnId)

  Create one or more Connected Isochronous Streams (CISes) using the connections identified by the ACL connection handles.
- void DmCisAccept (uint16_t handle)

  Inform the Controller to accept the request for the Connected Isochronous Stream (CIS) that is identified by the connection handle.
- void DmCisReject (uint16_t handle, uint8_t reason)

  Inform the Controller to reject the request for the Connected Isochronous Stream (CIS) that is identified by the connection handle.
- void DmCisClose (uint16_t handle, uint8_t reason)

  Close the Connected Isochronous Stream (CIS) connection with the given handle.
- uint8_t DmCisIdByHandle (uint16_t handle)

  For internal use only. Find the Connected Isochronous Stream (CIS) ID with matching handle.
- uint16_t DmCisHandleById (uint8_t cigId, uint8_t cisId)

  For internal use only. Find the Connected Isochronous Stream (CIS) handle with matching CIG and CIS identifiers.
- bool_t DmCisConnInUse (uint16_t handle)

For internal use only. Return TRUE if the Connected Isochronous Stream (CIS) connection is in use.

- uint8_t DmCisConnRole (uint16_t handle)

  For internal use only. Return the CIS connection role indicating master or slave.
- bool_t DmCisCigInUse (uint8_t cigId)

  For internal use only. Return TRUE if Connected Isochronous Group (CIG) is in use.
- bool_t DmCisInUse (uint8_t cigId, uint8_t cisId)

  For internal use only. Return TRUE if the Connected Isochronous Stream (CIS) connection is in use.

DM BIS Functions

Functions for forming and managing Broadcast Isochronous Stream (BIS) streams and synchronization.

- void DmBisSlaveInit (void)

  Initialize DM BIS manager for operation as slave.
- void DmBigStart (uint8_t bigHandle, uint8_t advHandle, uint8_t numBis, uint32_t sduInterUsec, uint16_t maxSdu, uint16_t mtlMs, uint8_t rtn)

  Start a Broadcast Isochronous Group (BIG) with one or more Broadcast Isochronous Streams (BISes).
- void DmBigStop (uint8_t bigHandle, uint8_t reason)

  Stop a Broadcast Isochronous Group (BIG) identified for the given handle.
- bool_t DmBisInUse (uint16_t handle)

  For internal use only. Return TRUE if the BIS is in use.
- void DmBigSetPhy (uint8_t bigHandle, uint8_t phyBits)

  Set the PHYs used for transmission of PDUs of Broadcast Isochronous Streams (BISes) in Broadcast Isochronous Group (BIG).
- void DmBigSetPackingFraming (uint8_t bigHandle, uint8_t packing, uint32_t framing)

  Set the packing scheme and framing format for the given Broadcast Isochronous Group (BIG).
- void DmBigSetBcastCode (uint8_t bigHandle, bool_t encrypt, bool_t authen, uint8_t *pBcastCode)

  Set the Broadcast Code for the given Broadcast Isochronous Group (BIG).
- void DmBigSetSecLevel (uint8_t bigHandle, uint8_t secLevel)

  Set the security level of the LE Security Mode 3 for the given Broadcast Isochronous Group (BIG).
- uint8_t DmBigGetSecLevel (uint16_t handle)

  Get the security level of the LE Security Mode 3 for the given Broadcast Isochronous Group (BIG) connection handle.

DM Isochronous (ISO) Functions

Functions for setting up and managing isochronous data path between the Host and the Controller.

- void DmIsoInit (void)

  Initialize DM ISO manager.
- void DmIsoRegister (hciIsoCback_t cisCback, hciIsoCback_t bisCback)

  Register CIS and BIS callbacks for the HCI ISO data path.
- void DmIsoDataPathSetup (HciIsoSetupDataPath_t *pDataPathParam)

  Setup the isochronous data path between the Host and the Controller for an established Connected Isochronous Stream (CIS) or Broadcast Isochronous Stream (BIS) identified by the connection handle parameter.
- void DmIsoDataPathRemove (uint16_t handle, uint8_t directionBits)

  Remove the input and/or output data path(s) associated with a Connected Isochronous Stream (CIS) or Broadcast Isochronous Stream (BIS) identified by the connection handle parameter.
- void DmDataPathConfig (HciConfigDataPath_t *pDataPathParam)

  Request the Controller to configure the data transport path in a given direction between the Controller and the Host.
- void DmReadLocalSupCodecs (void)

  Read a list of the codecs supported by the Controller, as well as vendor specific codecs, which are defined by an individual manufacturer.
- void DmReadLocalSupCodecCap (HciReadLocalSupCodecCaps_t *pCodecParam)

  Read a list of codec capabilities supported by the Controller for a given codec.
- void DmReadLocalSupCtrDly (HciReadLocalSupControllerDly_t *pDelayParam)

  Read the range of supported Controller delays for the codec specified by Codec ID on a given transport type specified by Logical Transport Type, in the direction specified by Direction, and with the codec configuration specified by Codec Configuration.

- void DmSendIsoData (uint16_t handle, uint16_t len, uint8_t *pData)
  Send ISO Data packet.

DM PHY Control Functions

Functions for setting PHY preferences.

- void DmSetDefaultPhy (uint8_t allPhys, uint8_t txPhys, uint8_t rxPhys)
  Set the preferred values for the transmitter PHY and receiver PHY for all subsequent connections.
- void DmReadPhy (dmConnId_t connId)
  Read the current transmitter PHY and receiver PHY for a given connection.
- void DmSetPhy (dmConnId_t connId, uint8_t allPhys, uint8_t txPhys, uint8_t rxPhys, uint16_t phyOptions)
  Set the PHY preferences for a given connection.
- void DmPhyInit (void)
  Initialize DM PHY.

DM Device Functions

Device control functions

- void DmDevReset (void)
  Reset the device.
- void DmDevSetRandAddr (uint8_t *pAddr)
  Set the random address to be used by the local device.
- void DmDevWhiteListAdd (uint8_t addrType, uint8_t *pAddr)
  Add a peer device to the white list. Note that this function cannot be called while advertising, scanning, or connecting with white list filtering active.
- void DmDevWhiteListRemove (uint8_t addrType, uint8_t *pAddr)
  Remove a peer device from the white list. Note that this function cannot be called while advertising, scanning, or connecting with white list filtering active.
- void DmDevWhiteListClear (void)
  Clear the white list. Note that this function cannot be called while advertising, scanning, or connecting with white list filtering active.
- bool_t DmDevSetFilterPolicy (uint8_t mode, uint8_t policy)
  Set the Advertising, Scanning or Initiator filter policy.
- bool_t DmDevSetExtFilterPolicy (uint8_t advHandle, uint8_t mode, uint8_t policy)
  Set the Advertising filter policy for the given advertising, Scanning or Initiator filter policy.
- void DmDevVsInit (uint8_t param)
  Vendor-specific controller initialization function.

DM Security Functions

Functions for accessing and controlling security configuration of device.

- void DmSecInit (void)
  Initialize DM security.
- void DmSecLescInit (void)
  Initialize DM LE Secure Connections security.
- void DmSecPairReq (dmConnId_t connId, uint8_t oob, uint8_t auth, uint8_t iKeyDist, uint8_t rKeyDist)
  This function is called by a master device to initiate pairing.
- void DmSecPairRsp (dmConnId_t connId, uint8_t oob, uint8_t auth, uint8_t iKeyDist, uint8_t rKeyDist)
  This function is called by a slave device to proceed with pairing after a DM_SEC_PAIR_IND event is received.
- void DmSecCancelReq (dmConnId_t connId, uint8_t reason)
  This function is called to cancel the pairing process.
- void DmSecAuthRsp (dmConnId_t connId, uint8_t authDataLen, uint8_t *pAuthData)
  This function is called in response to a DM_SEC_AUTH_REQ_IND event to provide PIN or OOB data during pairing.
- void DmSecSlaveReq (dmConnId_t connId, uint8_t auth)
  This function is called by a slave device to request that the master initiates pairing or link encryption.
- void DmSecEncryptReq (dmConnId_t connId, uint8_t secLevel, dmSecLtk_t *pLtk)

This function is called by a master device to initiate link encryption.

- void DmSecLtkRsp (dmConnId_t connId, bool_t keyFound, uint8_t secLevel, uint8_t *pKey)

  This function is called by a slave in response to a DM_SEC_LTK_REQ_IND event to provide the long term key used for encryption.
- void DmSecSetLocalCsrk (uint8_t *pCsrk)

  This function sets the local CSRK used by the device.
- void DmSecSetLocalIrk (uint8_t *pIrk)

  This function sets the local IRK used by the device.
- void DmSecGenerateEccKeyReq (void)

  This function generates an ECC key for use with LESC security.
- void DmSecSetEccKey (secEccKey_t *pKey)

  This function sets the ECC key for use with LESC security.
- secEccKey_t * DmSecGetEccKey (void)

  This function gets the local ECC key for use with LESC security.
- void DmSecSetDebugEccKey (void)

  This function sets the ECC key for use with LESC security to standard debug keys values.
- void DmSecSetOob (dmConnId_t connId, dmSecLescOobCfg_t *pConfig)

  This function configures the DM to use OOB pairing for the given connection. The pRand and pConfirm contain the Random and Confirm values exchanged via out of band methods.
- void DmSecCalcOobReq (uint8_t *pRand, uint8_t *pPubKeyX)

  This function calculates the local random and confirm values used in LESC OOB pairing. The operation's result is posted as a DM_SEC_CALC_OOB_IND event to the application's DM callback handler. The local rand and confirm values are exchanged with the peer via out-of-band (OOB) methods and passed into the DmSecSetOob after DM_CONN_OPEN_IND.
- void DmSecCompareRsp (dmConnId_t connId, bool_t valid)

  This function is called by the application in response to a DM_SEC_COMPARE_IND event. The valid parameter indicates if the compare value of the DM_SEC_COMPARE_IND was valid.
- uint32_t DmSecGetCompareValue (uint8_t *pConfirm)

  This function returns the 6-digit compare value for the specified 128-bit confirm value.

DM Internal Functions

Functions called internally by the stack.

- uint8_t DmLlAddrType (uint8_t addrType)

  Map an address type to a type used by LL.
- uint8_t DmHostAddrType (uint8_t addrType)

  Map an address type to a type used by Host.
- uint16_t DmSizeOfEvt (dmEvt_t *pDmEvt)

  Return size of a DM callback event.
- void DmL2cConnUpdateCnf (uint16_t handle, uint16_t reason)

  For internal use only. L2C calls this function to send the result of an L2CAP connection update response to DM.
- void DmL2cCmdRejInd (uint16_t handle, uint16_t result)

  For internal use only. L2C calls this function to send the result of an L2CAP Command Reject up to the application.
- void DmL2cConnUpdateInd (uint8_t identifier, uint16_t handle, hciConnSpec_t *pConnSpec)

  For internal use only. L2C calls this function when it receives a connection update request from a peer device.
- dmConnId_t DmConnIdByHandle (uint16_t handle)

  For internal use only. Find the connection ID with matching handle.
- bool_t DmConnInUse (dmConnId_t connId)

  For internal use only. Return TRUE if the connection is in use.
- uint8_t DmConnActiveCount (void)

  *Count active connections *.*
- uint8_t DmConnPeerAddrType (dmConnId_t connId)

  For internal use only. Return the peer address type.
- uint8_t * DmConnPeerAddr (dmConnId_t connId)

  For internal use only. Return the peer device address.
- uint8_t DmConnLocalAddrType (dmConnId_t connId)

  For internal use only. Return the local address type.
- uint8_t * DmConnLocalAddr (dmConnId_t connId)

  For internal use only. Return the local address.

- uint8_t * DmConnPeerRpa (dmConnId_t connId)
  For internal use only. Return the peer resolvable private address (RPA).
- uint8_t * DmConnLocalRpa (dmConnId_t connId)
  For internal use only. Return the local resolvable private address (RPA).
- uint8_t DmConnSecLevel (dmConnId_t connId)
  For internal use only. Return the security level of the connection.
- void DmSmpEncryptReq (dmConnId_t connId, uint8_t secLevel, uint8_t *pKey)
  For internal use only. This function is called by SMP to request encryption.
- void DmSmpCbackExec (dmEvt_t *pDmEvt)
  For internal use only. Execute DM callback from SMP procedures.
- uint8_t * DmSecGetLocalCsrk (void)
  For internal use only. This function gets the local CSRK used by the device.
- uint8_t * DmSecGetLocalIrk (void)
  For internal use only. This function gets the local IRK used by the device.
- void DmReadRemoteFeatures (dmConnId_t connId)
  For internal use only. Read the features of the remote device.
- void DmReadRemoteVerInfo (dmConnId_t connId)
  Read the version info of the remote device.
- void DmDisableSlaveLatency (dmConnId_t connId, bool_t disabled)
  Disable Slave Latency.
- void DmOverruleRemoteMaxRxOctetsAndTime (dmConnId_t connId, uint16_t maxRxOctetsRemote, uint16_t maxRxTimeRemote)
  Over rule Remote Maximum Rx octets.
- void HciVsdSetDeviceAddress (uint8_t *pAddr)
  Set device address.
- void HciVsdSetTransmitPower (int8_t transmitPower)
  Set transmit power.
- void HciCmndVsdSetLeMetaVSDEvent (uint8_t event)
  Set event notification bit.
- void HciCmndVsdResetLeMetaVSDEvent (uint8_t event)
  Reset event notification bit.

DM Callback Events

Events handled by the DM state machine.

- #define DM_CBACK_START 0x20
  DM callback event starting value.
- #define DM_CBACK_END DM_VENDOR_SPEC_IND
  DM callback event ending value.
- enum {
  DM_RESET_CMPL_IND = DM_CBACK_START,
  DM_ADV_START_IND,
  DM_ADV_STOP_IND,
  DM_ADV_NEW_ADDR_IND,
  DM_SCAN_START_IND,
  DM_SCAN_STOP_IND,
  DM_SCAN_REPORT_IND,
  DM_CONN_OPEN_IND,
  DM_CONN_CLOSE_IND,
  DM_CONN_UPDATE_IND,
  DM_SEC_PAIR_CMPL_IND,
  DM_SEC_PAIR_FAIL_IND,
  DM_SEC_ENCRYPT_IND,
  DM_SEC_ENCRYPT_FAIL_IND,

DM_SEC_AUTH_REQ_IND,
DM_SEC_KEY_IND,
DM_SEC_LTK_REQ_IND,
DM_SEC_PAIR_IND,
DM_SEC_SLAVE_REQ_IND,
DM_SEC_CALC_OOB_IND,
DM_SEC_ECC_KEY_IND,
DM_SEC_COMPARE_IND,
DM_SEC_KEYPRESS_IND,
DM_PRIV_RESOLVED_ADDR_IND,
DM_PRIV_GENERATE_ADDR_IND,
DM_CONN_READ_RSSI_IND,
DM_PRIV_ADD_DEV_TO_RES_LIST_IND,
DM_PRIV_REM_DEV_FROM_RES_LIST_IND,
DM_PRIV_CLEAR_RES_LIST_IND,
DM_PRIV_READ_PEER_RES_ADDR_IND,
DM_PRIV_READ_LOCAL_RES_ADDR_IND,
DM_PRIV_SET_ADDR_RES_ENABLE_IND,
DM_REM_CONN_PARAM_REQ_IND,
DM_CONN_DATA_LEN_CHANGE_IND,
DM_CONN_WRITE_AUTH_TO_IND,
DM_CONN_AUTH_TO_EXPIRED_IND,
DM_PHY_READ_IND,
DM_PHY_SET_DEF_IND,
DM_PHY_UPDATE_IND,
DM_ADV_SET_START_IND,
DM_ADV_SET_STOP_IND,
DM_SCAN_REQ_RCVD_IND,
DM_EXT_SCAN_START_IND,
DM_EXT_SCAN_STOP_IND,
DM_EXT_SCAN_REPORT_IND,
DM_PER_ADV_SET_START_IND,
DM_PER_ADV_SET_STOP_IND,
DM_PER_ADV_SYNC_EST_IND,
DM_PER_ADV_SYNC_EST_FAIL_IND,
DM_PER_ADV_SYNC_LOST_IND,
DM_PER_ADV_SYNC_TRSF_EST_IND,
DM_PER_ADV_SYNC_TRSF_EST_FAIL_IND,
DM_PER_ADV_SYNC_TRSF_IND,
DM_PER_ADV_SET_INFO_TRSF_IND,
DM_PER_ADV_REPORT_IND,
DM_REMOTE_FEATURES_IND,
DM_READ_REMOTE_VER_INFO_IND,
DM_CONN_IQ_REPORT_IND,
DM_CTE_REQ_FAIL_IND,
DM_CONN_CTE_RX_SAMPLE_START_IND,
DM_CONN_CTE_RX_SAMPLE_STOP_IND,
DM_CONN_CTE_TX_CFG_IND,
DM_CONN_CTE_REQ_START_IND,
DM_CONN_CTE_REQ_STOP_IND,
DM_CONN_CTE_RSP_START_IND,
DM_CONN_CTE_RSP_STOP_IND,
DM_READ_ANTENNA_INFO_IND,
DM_CIS_CIG_CONFIG_IND,
DM_CIS_CIG_REMOVE_IND,
DM_CIS_REQ_IND,
DM_CIS_OPEN_IND,
DM_CIS_CLOSE_IND,

DM_REQ_PEER_SCA_IND,
DM_ISO_DATA_PATH_SETUP_IND,
DM_ISO_DATA_PATH_REMOVE_IND,
DM_DATA_PATH_CONFIG_IND,
DM_READ_LOCAL_SUP_CODECS_IND,
DM_READ_LOCAL_SUP_CODEC_CAP_IND,
DM_READ_LOCAL_SUP_CTR_DLY_IND,
DM_BIG_START_IND,
DM_BIG_STOP_IND,
DM_BIG_SYNC_EST_IND,
DM_BIG_SYNC_EST_FAIL_IND,
DM_BIG_SYNC_LOST_IND,
DM_BIG_SYNC_STOP_IND,
DM_BIG_INFO_ADV_REPORT_IND,
DM_L2C_CMD_REJ_IND,
DM_ERROR_IND,
DM_HW_ERROR_IND,
DM_VENDOR_SPEC_IND }

DM callback events.

3.1.1 Detailed Description

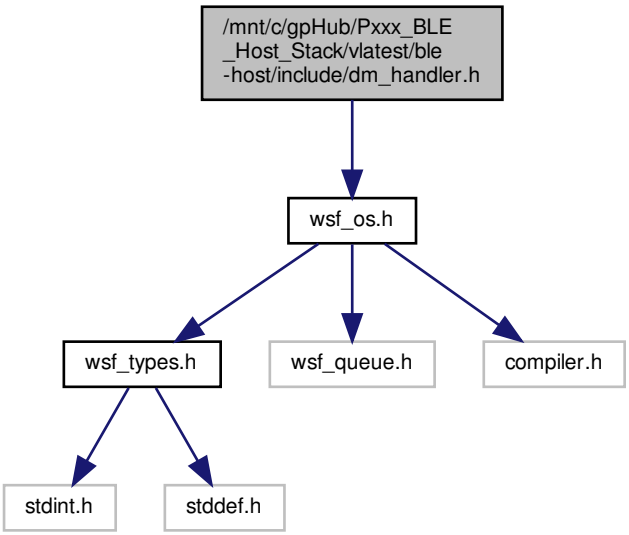
Device Manager subsystem API.

3.2 /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_handler.h File Reference

Interface to DM event handler.

```
#include "wsf_os.h"
```

Include dependency graph for dm_handler.h:



Functions

DM Event Handling

Message passing interface to DM from other tasks through WSF.

- void DmHandlerInit (wsfHandlerId_t handlerId)
  DM handler init function called during system initialization.
- void DmHandler (wsfEventMask_t event, wsfMsgHdr_t *pMsg)
  WSF event handler for DM.

3.2.1 Detailed Description

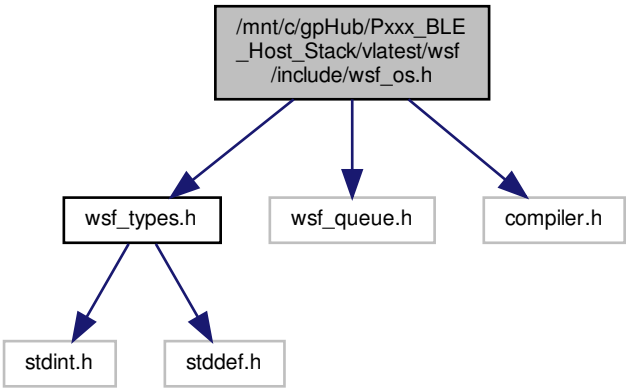
Interface to DM event handler.

3.3 /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/wsf/include/wsf_os.h File Reference

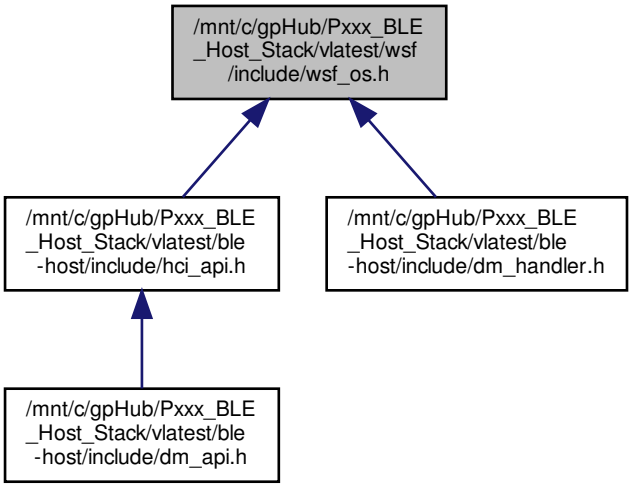
Software foundation OS API.

```
#include "wsf_types.h"
#include "wsf_queue.h"
#include "compiler.h"
```

Include dependency graph for wsf_os.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct wsfMsgHdr_t

  Common message structure passed to event handler.

Macros

- #define WSF_OS_DIAG FALSE

  OS Diagnostics.
- #define WSF_TASK_FROM_ID(handlerID) (((handlerID) >> 4) & 0x0F)

  Derive task from handler ID.
- #define WSF_HANDLER_FROM_ID(handlerID) ((handlerID) & 0x0F)

  Derive handler from handler ID.
- #define WSF_INVALID_TASK_ID 0xFF

  Invalid Task Identifier.
- #define WSF_OS_GET_ACTIVE_HANDLER_ID() WSF_INVALID_TASK_ID

  Get Diagnostic Task Identifier.

WSF Task Events

- #define WSF_MSG_QUEUE_EVENT 0x01

  Message queued for event handler.
- #define WSF_TIMER_EVENT 0x02

  Timer expired for event handler.
- #define WSF_HANDLER_EVENT 0x04

  Event set for event handler.

Typedefs

- typedef uint8_t wsfHandlerId_t

  Event handler ID data type.
- typedef uint16_t wsfEventMask_t

  Event handler event mask data type.
- typedef wsfHandlerId_t wsfTaskId_t

  Task ID data type.
- typedef uint8_t wsfTaskEvent_t

  Task event mask data type.
- typedef bool_t(* WsfOsIdleHandler_t) (void)

  Idle check function.
- typedef void(* wsfEventHandler_t) (wsfEventMask_t event, wsfMsgHdr_t *pMsg)

  Event handler callback function.

Functions

- void WsfSetEvent (wsfHandlerId_t handlerId, wsfEventMask_t event)

  Set an event for an event handler.
- void WsfTaskLock (void)

  Lock task scheduling.
- void WsfTaskUnlock (void)

  Unlock task scheduling.
- void WsfTaskSetReady (wsfHandlerId_t handlerId, wsfTaskEvent_t event)

  Set the task used by the given handler as ready to run.
- wsfQueue_t * WsfTaskMsgQueue (wsfHandlerId_t handlerId)

  Return the task message queue used by the given handler.
- wsfHandlerId_t WsfOsSetNextHandler (wsfEventHandler_t handler)

  Set the next WSF handler function in the WSF OS handler array. This function should only be called as part of the OS initialization procedure.
- void WsfOsInit (void)

  Initialize OS control structure.
- bool_t WsfOsReadyToSleep (void)

  Check if WSF is ready to sleep.
- void WsfOsDispatcher (void)

  Event dispatched. Designed to be called repeatedly from infinite loop.
- void WsfOsEnterMainLoop (void)

  OS starts main loop.
- void WsfOsRegisterIdleTask (WsfOsIdleHandler_t func)

  Register service check functions.

Variables

- wsfHandlerId_t WsfActiveHandler

  Diagnostic Task Identifier.

3.3.1 Detailed Description

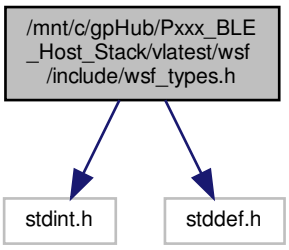
Software foundation OS API.

3.4 /mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/wsf/include/wsf_types.h File Reference

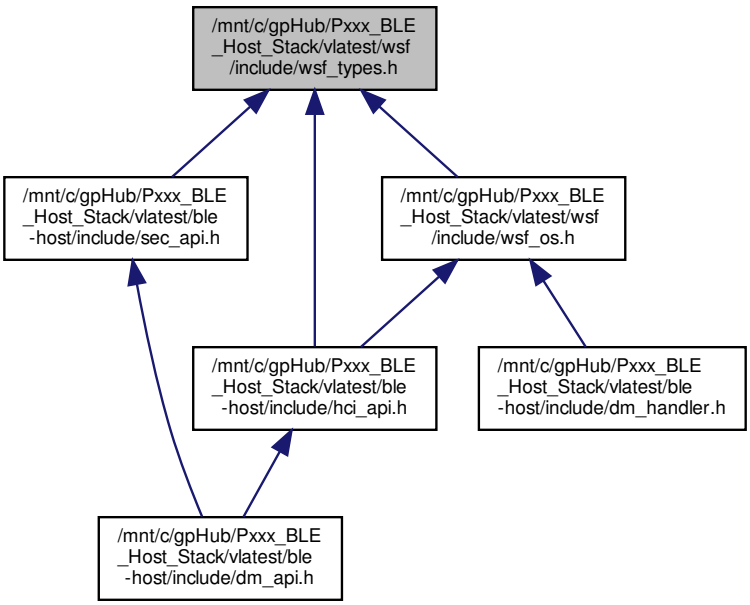
Platform-independent data types.

```
#include <stdint.h>
#include <stddef.h>
```

Include dependency graph for wsf_types.h:



This graph shows which files directly or indirectly include this file:



Macros

Integer Data Types

- #define **bool_t** uint8_t
- #define **FALSE** 0
- #define **TRUE** (!FALSE)
- #define **UINT64_C**(x) x##ULL
- #define **UINT32_C**(x) x##UL
- #define **UINT8_C**(x) (x)

3.4.1 Detailed Description

Platform-independent data types.

Index

/mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_api.h, 155
/mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/ble-host/include/dm_handler.h, 177
/mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/wsf/include/wsf_os.h, 179
/mnt/c/gpHub/Pxxx_BLE_Host_Stack/vlatest/wsf/include/wsf_types.h, 182

DM_RAND_ADDR_RPA
    Device Manager API, 31
DM_RAND_ADDR_SA
    Device Manager API, 31
Device Manager API, 6
    DM_RAND_ADDR_RPA, 31
    DM_RAND_ADDR_SA, 31
    DmAddDeviceToPerAdvList, 63
    DmAdvClearAdvSets, 40
    DmAdvConfig, 38
    DmAdvIncTxPwr, 45
    DmAdvInit, 37
    DmAdvModeExt, 38
    DmAdvModeLeg, 37
    DmAdvOmitAdvAddr, 44
    DmAdvRemoveAdvSet, 40
    DmAdvScanReqNotifEnable, 46
    DmAdvSetAdValue, 42
    DmAdvSetAddrType, 42
    DmAdvSetChannelMap, 41
    DmAdvSetData, 38
    DmAdvSetFragPref, 46
    DmAdvSetInterval, 41
    DmAdvSetName, 43
    DmAdvSetPhyParam, 45
    DmAdvSetRandAddr, 40
    DmAdvSetSid, 46
    DmAdvStart, 39
    DmAdvStop, 39
    DmAdvUseLegacyPdu, 44
    DmBigGetSecLevel, 92
    DmBigSetBcastCode, 91
    DmBigSetPackingFraming, 90
    DmBigSetPhy, 90
    DmBigSetSecLevel, 91
    DmBigStart, 89
    DmBigStop, 89
    DmBigSyncGetSecLevel, 63
    DmBigSyncSetBcastCode, 62
    DmBigSyncSetSecLevel, 62
    DmBigSyncStart, 61
    DmBigSyncStop, 61
    DmBisInUse, 90
    DmBisMasterInit, 63
    DmBisSlaveInit, 88
    DmBisSyncInUse, 62
    DmCisAccept, 85
    DmCisCigConfig, 84
    DmCisCigInUse, 88
    DmCisCigRemove, 84
    DmCisCigSetPackingFraming, 83
    DmCisCigSetSca, 82
    DmCisCigSetSduInterval, 82
    DmCisCigSetTransLatInterval, 83
    DmCisClose, 86
    DmCisConnInUse, 87
    DmCisConnRole, 87
    DmCisHandleById, 86
    DmCisIdByHandle, 86
    DmCisInUse, 88
    DmCisInit, 81
    DmCisMasterInit, 81
    DmCisOpen, 84
    DmCisReject, 85
    DmCisSlaveInit, 81
    DmClearPerAdvList, 64
    DmConnAccept, 74
    DmConnActiveCount, 111
    DmConnCheckIdle, 78
    DmConnClose, 73
    DmConnCteGetReqState, 70
    DmConnCteGetRspState, 70
    DmConnCteInit, 56
    DmConnCteReqStart, 68
    DmConnCteReqStop, 69
    DmConnCteRspStart, 69
    DmConnCteRspStop, 69
    DmConnCteRxSampleStart, 67
    DmConnCteRxSampleStop, 67
    DmConnCteTxConfig, 68
    DmConnIdByHandle, 110
    DmConnInUse, 110
    DmConnInit, 71
    DmConnLocalAddr, 112
    DmConnLocalAddrType, 112
    DmConnLocalRpa, 113
    DmConnMasterInit, 71
    DmConnOpen, 73
    DmConnPeerAddr, 111
    DmConnPeerAddrType, 111

DmConnPeerRpa, 112
DmConnReadRssi, 78
DmConnRegister, 72
DmConnRequestPeerSca, 81
DmConnRole, 80
DmConnSecLevel, 113
DmConnSetAddrType, 77
DmConnSetConnSpec, 75
DmConnSetDataLen, 79
DmConnSetIdle, 77
DmConnSetScanInterval, 75
DmConnSlaveInit, 72
DmConnUpdate, 74
DmDataPathConfig, 93
DmDevPrivInit, 43
DmDevPrivStart, 43
DmDevPrivStop, 44
DmDevReset, 97
DmDevSetExtFilterPolicy, 99
DmDevSetFilterPolicy, 98
DmDevSetRandAddr, 97
DmDevVsInit, 99
DmDevWhiteListAdd, 97
DmDevWhiteListClear, 98
DmDevWhiteListRemove, 98
DmDisableSlaveLatency, 115
DmExtAdvInit, 37
DmExtConnMasterInit, 71
DmExtConnSetConnSpec, 77
DmExtConnSetScanInterval, 75
DmExtConnSlaveInit, 72
DmExtMaxAdvDataLen, 49
DmExtScanInit, 55
DmFindAdType, 37
DmHostAddrType, 108
DmIsoDataPathRemove, 93
DmIsoDataPathSetup, 93
DmIsoInit, 92
DmIsoRegister, 92
DmL2cCmdRejInd, 109
DmL2cConnUpdateCnf, 109
DmL2cConnUpdateInd, 109
DmLlAddrType, 108
DmLlPrivEnabled, 55
DmOverruleRemoteMaxRxOctetsAndTime, 116
DmPastConfig, 66
DmPastDefaultConfig, 66
DmPastInit, 56
DmPastRptRcvEnable, 64
DmPastSetInfoTrsf, 65
DmPastSyncTrsf, 65
DmPerAdvConfig, 47
DmPerAdvEnabled, 49
DmPerAdvIncTxPwr, 49
DmPerAdvSetData, 47
DmPerAdvSetInterval, 48
DmPerAdvStart, 48
DmPerAdvStop, 48
DmPhyInit, 96
DmPrivAddDevToResList, 50
DmPrivClearResList, 52
DmPrivGenerateAddr, 55
DmPrivInit, 50
DmPrivReadLocalResolvableAddr, 53
DmPrivReadPeerResolvableAddr, 52
DmPrivRemDevFromResList, 51
DmPrivResolveAddr, 50
DmPrivSetAddrResEnable, 53
DmPrivSetPrivacyMode, 54
DmPrivSetResolvablePrivateAddrTimeout, 54
DmReadAntennaInfo, 70
DmReadLocalSupCodecCap, 94
DmReadLocalSupCodecs, 94
DmReadLocalSupCtrDly, 94
DmReadPhy, 95
DmReadRemoteFeatures, 115
DmReadRemoteVerInfo, 115
DmRegister, 36
DmRemoteConnParamReqNegReply, 79
DmRemoteConnParamReqReply, 79
DmRemoveDeviceFromPerAdvList, 64
DmScanInit, 55
DmScanModeExt, 56
DmScanModeLeg, 56
DmScanSetAddrType, 58
DmScanSetInterval, 57
DmScanStart, 57
DmScanStop, 57
DmSecAuthRsp, 101
DmSecCalcOobReq, 106
DmSecCancelReq, 101
DmSecCompareRsp, 107
DmSecEncryptReq, 103
DmSecGenerateEccKeyReq, 105
DmSecGetCompareValue, 107
DmSecGetEccKey, 105
DmSecGetLocalCsrk, 114
DmSecGetLocalIrk, 114
DmSecInit, 99
DmSecLescInit, 100
DmSecLtkRsp, 104
DmSecPairReq, 100
DmSecPairRsp, 100
DmSecSetDebugEccKey, 106
DmSecSetEccKey, 105
DmSecSetLocalCsrk, 104
DmSecSetLocalIrk, 104
DmSecSetOob, 106
DmSecSlaveReq, 103
DmSendIsoData, 95
DmSetDefaultPhy, 95
DmSetPhy, 96
DmSizeOfEvt, 108
DmSmpCbackExec, 114
DmSmpEncryptReq, 113
DmSyncEnabled, 60

DmSyncEncrypted, 60
DmSyncInitialRptEnable, 60
DmSyncSetEncrypt, 59
DmSyncStart, 58
DmSyncStop, 59
DmWriteAuthPayloadTimeout, 80
HciCmndVsdResetLeMetaVSDEvent, 117
HciCmndVsdSetLeMetaVSDEvent, 117
HciVsdSetDeviceAddress, 116
HciVsdSetTransmitPower, 117
DmAddDeviceToPerAdvList
Device Manager API, 63
DmAdvClearAdvSets
Device Manager API, 40
DmAdvConfig
Device Manager API, 38
DmAdvIncTxPwr
Device Manager API, 45
DmAdvInit
Device Manager API, 37
DmAdvModeExt
Device Manager API, 38
DmAdvModeLeg
Device Manager API, 37
dmAdvNewAddrIndEvt_t, 127
DmAdvOmitAdvAddr
Device Manager API, 44
DmAdvRemoveAdvSet
Device Manager API, 40
DmAdvScanReqNotifEnable
Device Manager API, 46
DmAdvSetAdValue
Device Manager API, 42
DmAdvSetAddrType
Device Manager API, 42
DmAdvSetChannelMap
Device Manager API, 41
DmAdvSetData
Device Manager API, 38
DmAdvSetFragPref
Device Manager API, 46
DmAdvSetInterval
Device Manager API, 41
DmAdvSetName
Device Manager API, 43
DmAdvSetPhyParam
Device Manager API, 45
DmAdvSetRandAddr
Device Manager API, 40
DmAdvSetSid
Device Manager API, 46
dmAdvSetStartEvt_t, 128
DmAdvStart
Device Manager API, 39
DmAdvStop
Device Manager API, 39
DmAdvUseLegacyPdu
Device Manager API, 44
DmBigGetSecLevel
Device Manager API, 92
DmBigSetBcastCode
Device Manager API, 91
DmBigSetPackingFraming
Device Manager API, 90
DmBigSetPhy
Device Manager API, 90
DmBigSetSecLevel
Device Manager API, 91
DmBigStart
Device Manager API, 89
DmBigStop
Device Manager API, 89
DmBigSyncGetSecLevel
Device Manager API, 63
DmBigSyncSetBcastCode
Device Manager API, 62
DmBigSyncSetSecLevel
Device Manager API, 62
DmBigSyncStart
Device Manager API, 61
DmBigSyncStop
Device Manager API, 61
DmBisInUse
Device Manager API, 90
DmBisMasterInit
Device Manager API, 63
DmBisSlaveInit
Device Manager API, 88
DmBisSyncInUse
Device Manager API, 62
dmCfg_t, 129
DmCisAccept
Device Manager API, 85
DmCisCigConfig
Device Manager API, 84
DmCisCigInUse
Device Manager API, 88
DmCisCigRemove
Device Manager API, 84
DmCisCigSetPackingFraming
Device Manager API, 83
DmCisCigSetSca
Device Manager API, 82
DmCisCigSetSduInterval
Device Manager API, 82
DmCisCigSetTransLatInterval
Device Manager API, 83
DmCisClose
Device Manager API, 86
DmCisConnInUse
Device Manager API, 87
DmCisConnRole
Device Manager API, 87
DmCisHandleById
Device Manager API, 86
DmCisIdByHandle

Device Manager API, 86
DmCisInUse
Device Manager API, 88
DmCisInit
Device Manager API, 81
DmCisMasterInit
Device Manager API, 81
DmCisOpen
Device Manager API, 84
DmCisReject
Device Manager API, 85
DmCisSlaveInit
Device Manager API, 81
DmClearPerAdvList
Device Manager API, 64
DmConnAccept
Device Manager API, 74
DmConnActiveCount
Device Manager API, 111
DmConnCheckIdle
Device Manager API, 78
DmConnClose
Device Manager API, 73
DmConnCteGetReqState
Device Manager API, 70
DmConnCteGetRspState
Device Manager API, 70
DmConnCteInit
Device Manager API, 56
DmConnCteReqStart
Device Manager API, 68
DmConnCteReqStop
Device Manager API, 69
DmConnCteRspStart
Device Manager API, 69
DmConnCteRspStop
Device Manager API, 69
DmConnCteRxSampleStart
Device Manager API, 67
DmConnCteRxSampleStop
Device Manager API, 67
DmConnCteTxConfig
Device Manager API, 68
DmConnIdByHandle
Device Manager API, 110
DmConnInUse
Device Manager API, 110
DmConnInit
Device Manager API, 71
DmConnLocalAddr
Device Manager API, 112
DmConnLocalAddrType
Device Manager API, 112
DmConnLocalRpa
Device Manager API, 113
DmConnMasterInit
Device Manager API, 71
DmConnOpen
Device Manager API, 73
DmConnPeerAddr
Device Manager API, 111
DmConnPeerAddrType
Device Manager API, 111
DmConnPeerRpa
Device Manager API, 112
DmConnReadRssi
Device Manager API, 78
DmConnRegister
Device Manager API, 72
DmConnRequestPeerSca
Device Manager API, 81
DmConnRole
Device Manager API, 80
DmConnSecLevel
Device Manager API, 113
DmConnSetAddrType
Device Manager API, 77
DmConnSetConnSpec
Device Manager API, 75
DmConnSetDataLen
Device Manager API, 79
DmConnSetIdle
Device Manager API, 77
DmConnSetScanInterval
Device Manager API, 75
DmConnSlaveInit
Device Manager API, 72
DmConnUpdate
Device Manager API, 74
DmDataPathConfig
Device Manager API, 93
DmDevPrivInit
Device Manager API, 43
DmDevPrivStart
Device Manager API, 43
DmDevPrivStop
Device Manager API, 44
DmDevReset
Device Manager API, 97
DmDevSetExtFilterPolicy
Device Manager API, 99
DmDevSetFilterPolicy
Device Manager API, 98
DmDevSetRandAddr
Device Manager API, 97
DmDevVsInit
Device Manager API, 99
DmDevWhiteListAdd
Device Manager API, 97
DmDevWhiteListClear
Device Manager API, 98
DmDevWhiteListRemove
Device Manager API, 98
DmDisableSlaveLatency
Device Manager API, 115
dmEvt_t, 130

DmExtAdvInit
Device Manager API, 37
DmExtConnMasterInit
Device Manager API, 71
DmExtConnSetConnSpec
Device Manager API, 77
DmExtConnSetScanInterval
Device Manager API, 75
DmExtConnSlaveInit
Device Manager API, 72
DmExtMaxAdvDataLen
Device Manager API, 49
DmExtScanInit
Device Manager API, 55
DmFindAdType
Device Manager API, 37
DmHandler
STACK_EVENT, 119
DmHandlerInit
STACK_EVENT, 119
DmHostAddrType
Device Manager API, 108
DmIsoDataPathRemove
Device Manager API, 93
DmIsoDataPathSetup
Device Manager API, 93
DmIsoInit
Device Manager API, 92
DmIsoRegister
Device Manager API, 92
dmL2cCmdRejEvt_t, 134
DmL2cCmdRejInd
Device Manager API, 109
DmL2cConnUpdateCnf
Device Manager API, 109
DmL2cConnUpdateInd
Device Manager API, 109
DmLlAddrType
Device Manager API, 108
DmLlPrivEnabled
Device Manager API, 55
DmOverruleRemoteMaxRxOctetsAndTime
Device Manager API, 116
DmPastConfig
Device Manager API, 66
DmPastDefaultConfig
Device Manager API, 66
DmPastInit
Device Manager API, 56
DmPastRptRcvEnable
Device Manager API, 64
DmPastSetInfoTrsf
Device Manager API, 65
DmPastSyncTrsf
Device Manager API, 65
DmPerAdvConfig
Device Manager API, 47
DmPerAdvEnabled
Device Manager API, 49
DmPerAdvIncTxPwr
Device Manager API, 49
DmPerAdvSetData
Device Manager API, 47
DmPerAdvSetInterval
Device Manager API, 48
dmPerAdvSetStartEvt_t, 135
dmPerAdvSetStopEvt_t, 136
DmPerAdvStart
Device Manager API, 48
DmPerAdvStop
Device Manager API, 48
DmPhyInit
Device Manager API, 96
DmPrivAddDevToResList
Device Manager API, 50
DmPrivClearResList
Device Manager API, 52
dmPrivGenAddrIndEvt_t, 137
DmPrivGenerateAddr
Device Manager API, 55
DmPrivInit
Device Manager API, 50
DmPrivReadLocalResolvableAddr
Device Manager API, 53
DmPrivReadPeerResolvableAddr
Device Manager API, 52
DmPrivRemDevFromResList
Device Manager API, 51
DmPrivResolveAddr
Device Manager API, 50
DmPrivSetAddrResEnable
Device Manager API, 53
DmPrivSetPrivacyMode
Device Manager API, 54
DmPrivSetResolvablePrivateAddrTimeout
Device Manager API, 54
DmReadAntennaInfo
Device Manager API, 70
DmReadLocalSupCodecCap
Device Manager API, 94
DmReadLocalSupCodecs
Device Manager API, 94
DmReadLocalSupCtrDly
Device Manager API, 94
DmReadPhy
Device Manager API, 95
DmReadRemoteFeatures
Device Manager API, 115
DmReadRemoteVerInfo
Device Manager API, 115
DmRegister
Device Manager API, 36
DmRemoteConnParamReqNegReply
Device Manager API, 79
DmRemoteConnParamReqReply
Device Manager API, 79

DmRemoveDeviceFromPerAdvList
Device Manager API, 64
dmRemoveIsoDataPathEvt_t, 138
DmScanInit
Device Manager API, 55
DmScanModeExt
Device Manager API, 56
DmScanModeLeg
Device Manager API, 56
DmScanSetAddrType
Device Manager API, 58
DmScanSetInterval
Device Manager API, 57
DmScanStart
Device Manager API, 57
DmScanStop
Device Manager API, 57
dmSecAuthReqIndEvt_t, 139
DmSecAuthRsp
Device Manager API, 101
DmSecCalcOobReq
Device Manager API, 106
DmSecCancelReq
Device Manager API, 101
dmSecCnfIndEvt_t, 140
DmSecCompareRsp
Device Manager API, 107
dmSecCsrk_t, 141
dmSecEncryptIndEvt_t, 141
DmSecEncryptReq
Device Manager API, 103
DmSecGenerateEccKeyReq
Device Manager API, 105
DmSecGetCompareValue
Device Manager API, 107
DmSecGetEccKey
Device Manager API, 105
DmSecGetLocalCsrk
Device Manager API, 114
DmSecGetLocalIrk
Device Manager API, 114
DmSecInit
Device Manager API, 99
dmSecIrk_t, 142
dmSecKey_t, 143
dmSecKeyIndEvt_t, 144
dmSecKeypressIndEvt_t, 146
DmSecLescInit
Device Manager API, 100
dmSecLescOobCfg_t, 147
dmSecLtk_t, 148
DmSecLtkRsp
Device Manager API, 104
dmSecOobCalcIndEvt_t, 149
dmSecPairCmplIndEvt_t, 150
dmSecPairIndEvt_t, 151
DmSecPairReq
Device Manager API, 100
DmSecPairRsp
Device Manager API, 100
DmSecSetDebugEccKey
Device Manager API, 106
DmSecSetEccKey
Device Manager API, 105
DmSecSetLocalCsrk
Device Manager API, 104
DmSecSetLocalIrk
Device Manager API, 104
DmSecSetOob
Device Manager API, 106
dmSecSlaveIndEvt_t, 152
DmSecSlaveReq
Device Manager API, 103
DmSendIsoData
Device Manager API, 95
DmSetDefaultPhy
Device Manager API, 95
DmSetPhy
Device Manager API, 96
dmSetupIsoDataPathEvt_t, 153
DmSizeOfEvt
Device Manager API, 108
DmSmpCbackExec
Device Manager API, 114
DmSmpEncryptReq
Device Manager API, 113
DmSyncEnabled
Device Manager API, 60
DmSyncEncrypted
Device Manager API, 60
DmSyncInitialRptEnable
Device Manager API, 60
DmSyncSetEncrypt
Device Manager API, 59
DmSyncStart
Device Manager API, 58
DmSyncStop
Device Manager API, 59
DmWriteAuthPayloadTimeout
Device Manager API, 80

GAP Device Manager (DM), 1

HciCmndVsdResetLeMetaVSDEvent
Device Manager API, 117
HciCmndVsdSetLeMetaVSDEvent
Device Manager API, 117
HciVsdSetDeviceAddress
Device Manager API, 116
HciVsdSetTransmitPower
Device Manager API, 117

STACK_EVENT, 119
DmHandler, 119
DmHandlerInit, 119

WSF_OS_API, 121

wsfEventHandler_t, 122
WsfOsDispatcher, 124
WsfOsInit, 124
WsfOsReadyToSleep, 124
WsfOsRegisterIdleTask, 125
WsfOsSetNextHandler, 124
WsfSetEvent, 123
WsfTaskMsgQueue, 123
WsfTaskSetReady, 123
WSF_TYPES, 126
wsfEventHandler_t
WSF_OS_API, 122
wsfMsgHdr_t, 154
WsfOsDispatcher
WSF_OS_API, 124
WsfOsInit
WSF_OS_API, 124
WsfOsReadyToSleep
WSF_OS_API, 124
WsfOsRegisterIdleTask
WSF_OS_API, 125
WsfOsSetNextHandler
WSF_OS_API, 124
WsfSetEvent
WSF_OS_API, 123
WsfTaskMsgQueue
WSF_OS_API, 123
WsfTaskSetReady
WSF_OS_API, 123