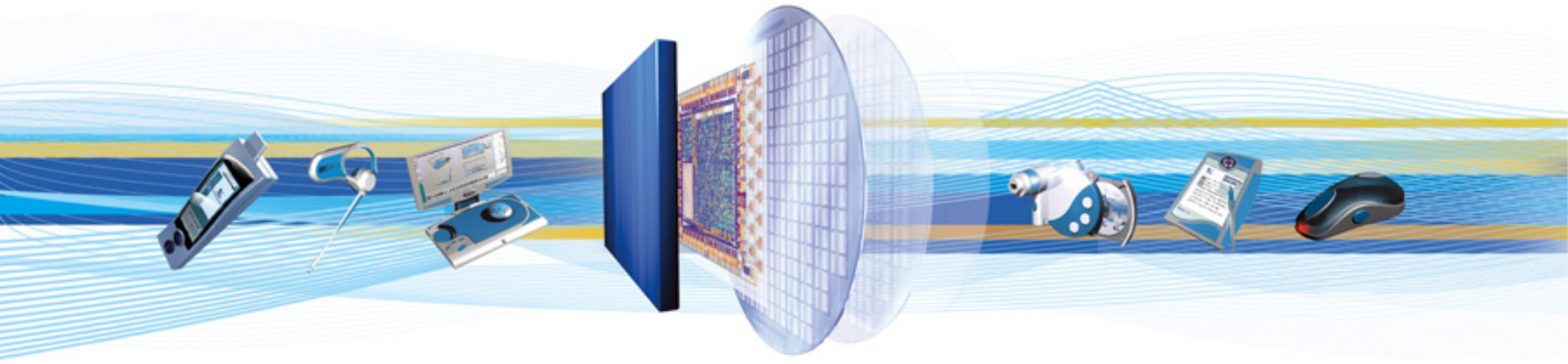
CSR Synergy Bluetooth 18.2.1

CM – Connection Manager

API Description

December 2011



Cambridge Silicon Radio Limited

Churchill House
Cambridge Business Park
Cowley Road
Cambridge CB4 0WZ
United Kingdom

Registered in England and Wales 3665875

Tel: +44 (0)1223 692000
Fax: +44 (0)1223 692001
www.csr.com

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction and Scope

This document describes the functionality and message interface provided by the Connection Manager (CM). The functions provided by the CM can be used for setting of the BlueCore device name, discovery etc.

1.2 Assumptions

The following assumptions and preconditions are made in the following:

- There is a secure and reliable transport between the profile part, i.e. the CM management and the application, for communication between the components

2 Description

2.1 Introduction

The CM module provides functions that may be used by various profiles and applications. It is an autonomous module, which provides functions for:

- Discovery of remote devices
- Changing the local device name
- Reading the local device address

2.2 Reference Model

The management module provides an abstraction layer to a subset of the low level Bluetooth® HCI interface.

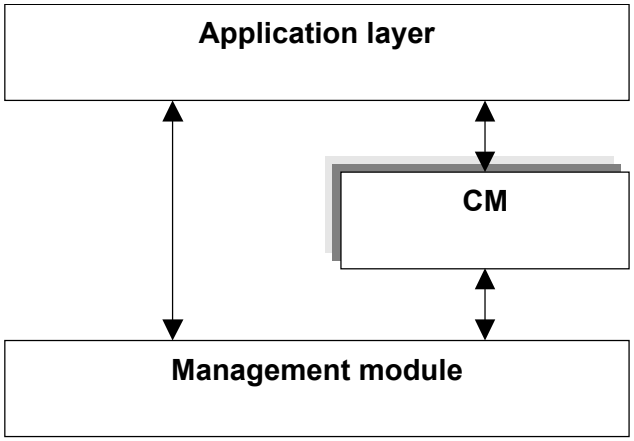


Figure 1: Reference model

The management module is also used by the CM.

2.3 Sequence Overview

Not applicable.

3 Interface Description

3.1 Application Layer API

3.1.1 Changing the Local Device Name

The local device can be assigned a Bluetooth® device name. The name is stored in the host controller by the CM.

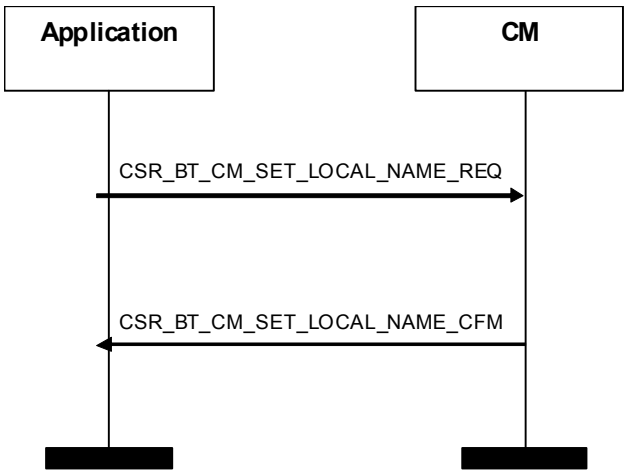


Figure 2: Change local name sequence

This name is revealed to remote devices upon request, e.g. as part of a device discovery sequence. The name should be set to an explanatory string by which the local Bluetooth® device can be identified.

3.1.2 Read the Local Bluetooth® Address

The local Bluetooth® address can be read.

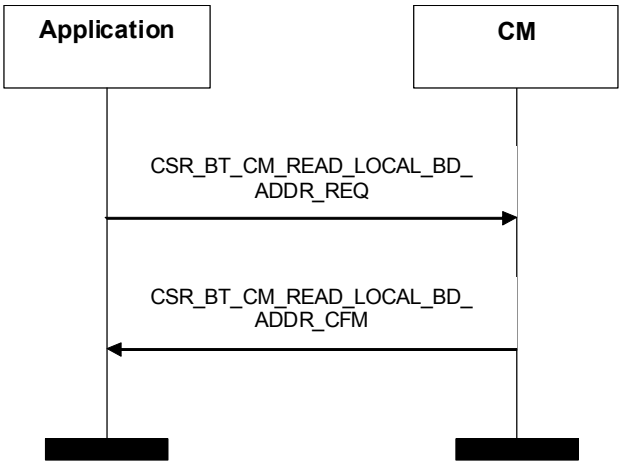


Figure 3: Read local device address sequence

3.1.3 Write Link Supervision Timeout

This command will write the value for the Link Supervision Timeout parameter for the device.

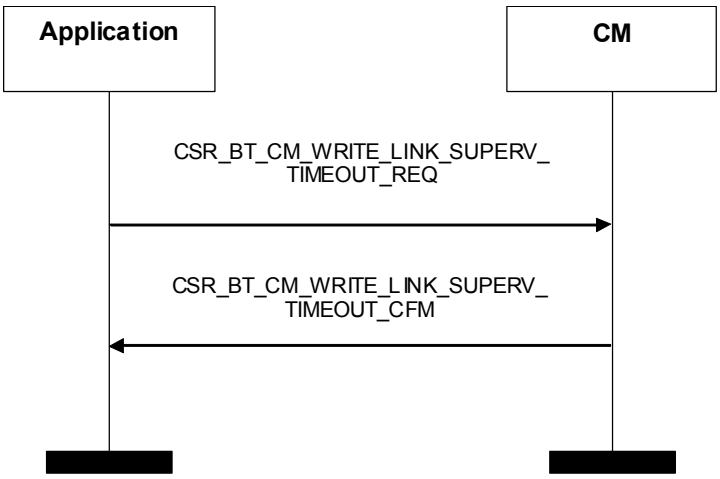


Figure 4: Write Link Supervision Timeout sequence

3.1.4 Read Remote Name

This command will request the remote (friendly) name of a remote device, which is identified by its Bluetooth Device Address.

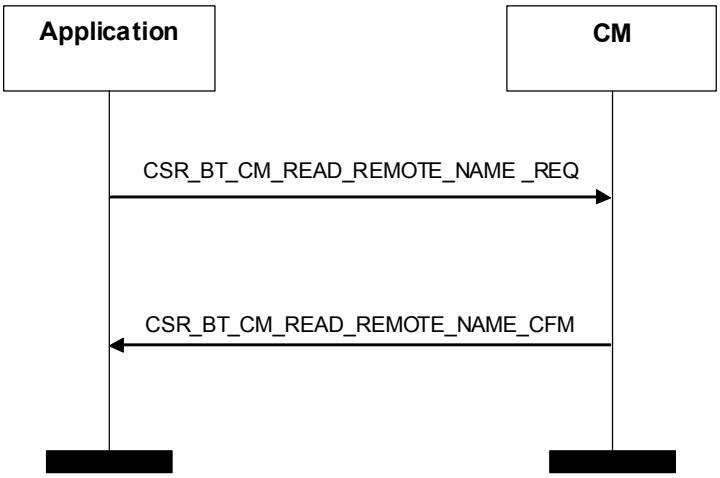


Figure 5: Read Remote Name sequence

3.1.5 Cancel Read Remote Name

This command will cancel a previously initiated read remote name procedure. The result code of CSR_BT_CM_READ_REMOTE_NAME_CFM will tell if the procedure was cancelled, or if it was completed before processing the cancel. Please note that no CFM signal will be received if there is no CSR_BT_CM_READ_REMOTE_NAME_REQ sent to the CM.

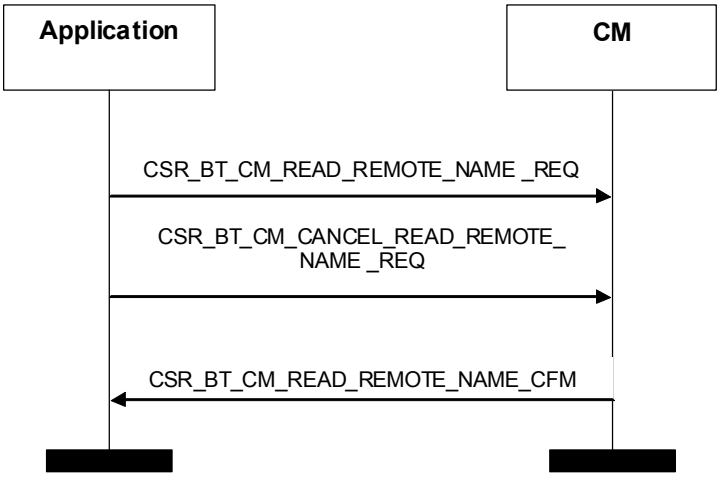


Figure 6: Read Remote Name sequence

3.1.6 Read Remote Version

This command will request to read the version of a remote device, which is identified by its Bluetooth Device Address.

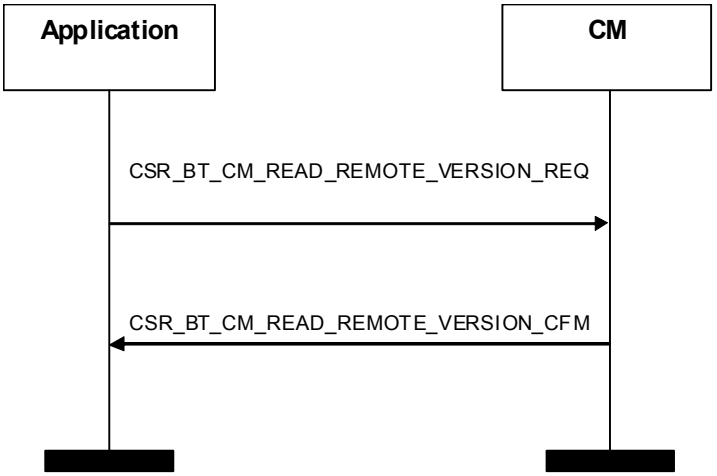


Figure 7: Read Remote Version sequence

3.1.7 Write Scan Enable Request

The Write Scan Enable command controls whether or not the local Bluetooth® device will periodically scan for page attempts and/or inquiry requests from other Bluetooth® devices.

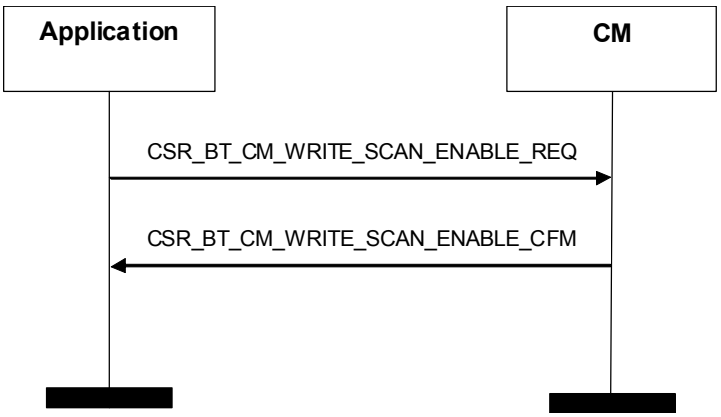


Figure 8: Write Scan Enable Request

3.1.8 Read Scan Enable Request

The Read Scan Enable command will read the value for the Scan Enable configuration parameter, which controls whether or not the Bluetooth device will periodically scan for page attempts and/or inquiry requests from other Bluetooth devices.

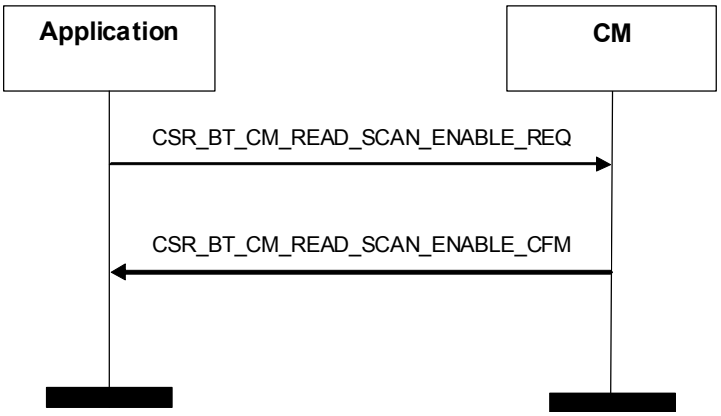


Figure 9: Read Scan Enable Request

3.1.9 Write Pagescan Settings

The Write Pagescan Settings command controls the timing of the pagescan process that runs if the Scan Enable configuration parameter has enabled pagescan.

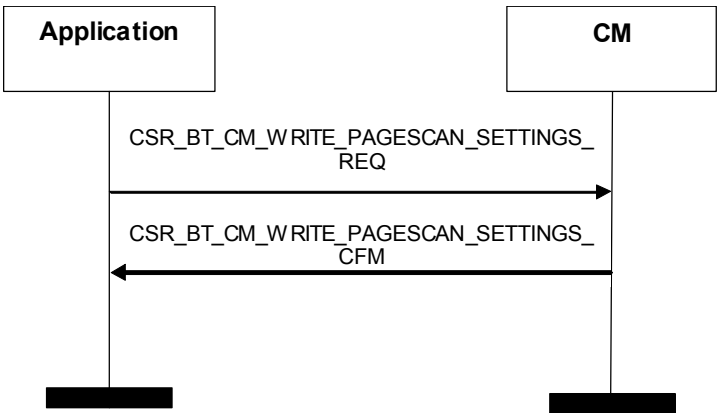


Figure 10: Write Pagescan Settings Request

3.1.10 Write Pagescan Type

The Write Pagescan Type command controls the type of pagescan that is performed if the Scan Enable configuration parameter has enabled pagescan.

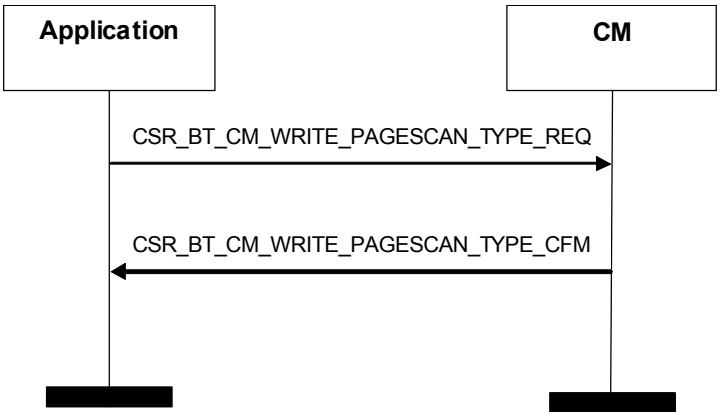


Figure 11: Write Pagescan Type Request

3.1.11 Write Inquiryscan Settings

The Write Inquiryscan Settings command controls the timing of the inquiryscan process that runs if the Scan Enable configuration parameter has enabled inquiryscan.

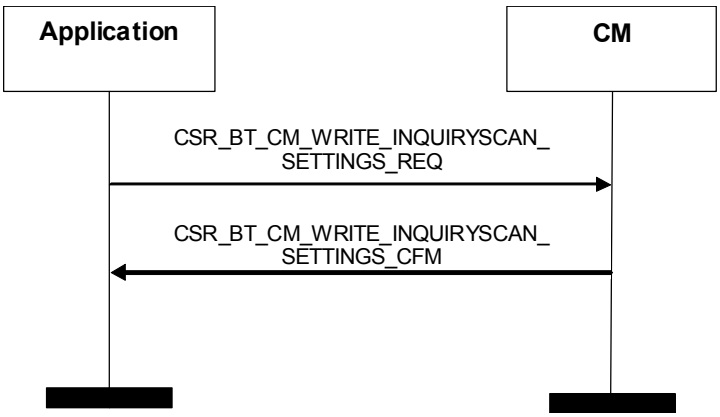


Figure 12: Write Inquiryscan Settings Request

3.1.12 Write Inquiryscan Type

The Write Inquiryscan Type command controls the type of inquiryscan that is performed if the Scan Enable configuration parameter has enabled inquiryscan.

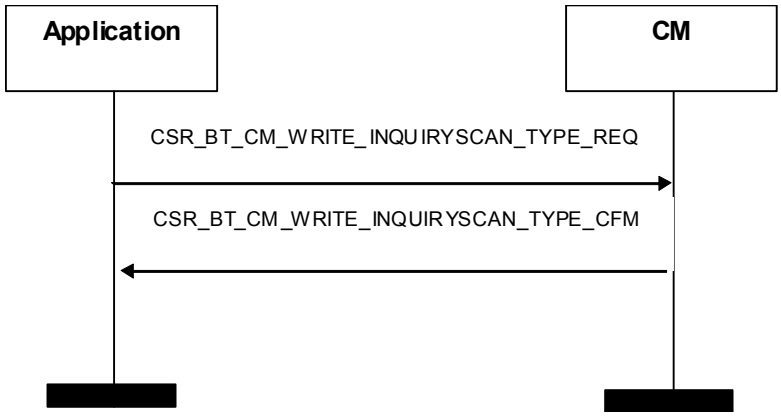


Figure 13: Write Inquiryscan Type Request

3.1.13 Connectable Request

The connectable procedure will write the value for the connectable parameter, which controls whether or not the calling process will be informed about rejected incoming RFCOMM connection attempts.

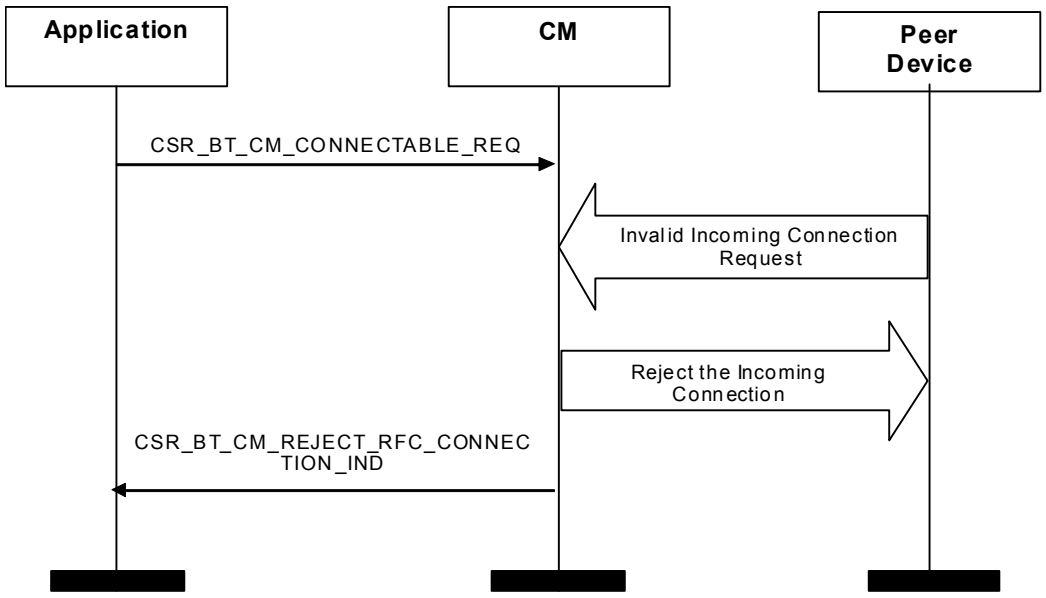


Figure 14: Connectable request

3.1.14 Device under Test Request

This command will set the BlueCore chip in Enable Device under Test Mode. To set the chip in this mode call the function CsrBtCmEnableDutModeReqSend located in csr_bt_cm_lib.h. Send the confirm message to the appHandle (task queue) parameter. The confirm message is CSR_BT_CM_ENABLE_DUT_MODE_CFM. There are two parameters to this message - the status of the operation and the step number indicating how many of the steps that are being handled. The enabled device under test consists of 3 steps in case of success.

To disable the device under test mode the chip needs to be reset and the CSR Synergy Bluetooth also needs to be restarted.

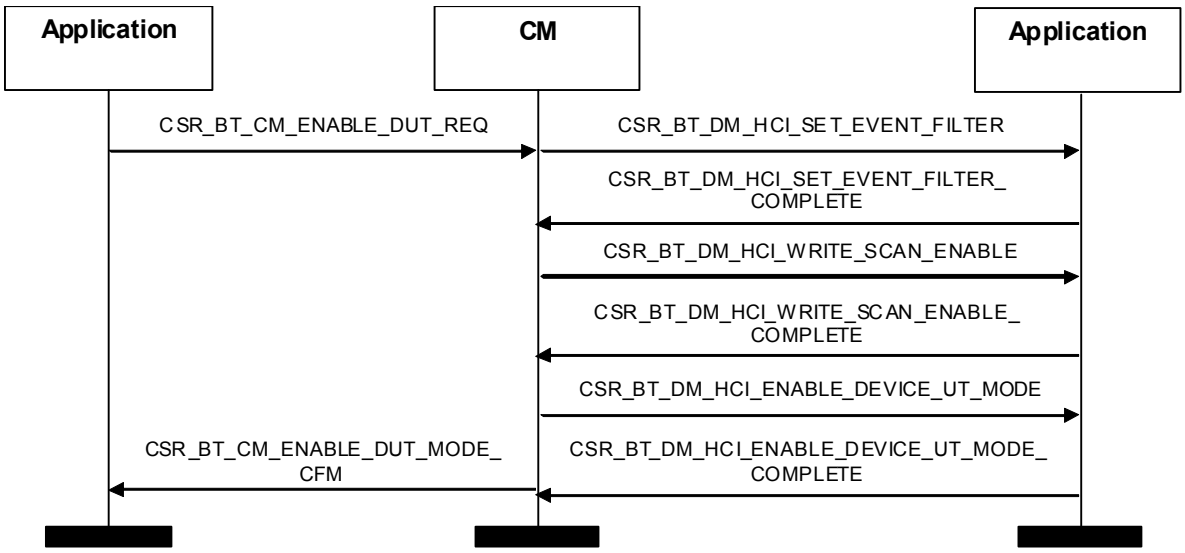


Figure 15: Enable device under test sequence

3.1.15 Device under Test Disable Request

Wjen the BlueCore chipis in Device under Test Mode and the higher layer wants to go back to normal operation mode, it shall call the function CsrBtCmDisableDutModeReqSend located in csr_bt_cm_lib.h. The CM will exit the Device under Test mode and send the confirm message to the appHandle (task queue) parameter. The confirm message is CSR_BT_CM_DISABLE_DUT_MODE_CFM.

To disable the device under test mode the chip needs to be reset. This operation is needed to ensure that the chip, the CM module and the higher layers are synchronized with regards to the Device under Test feature.

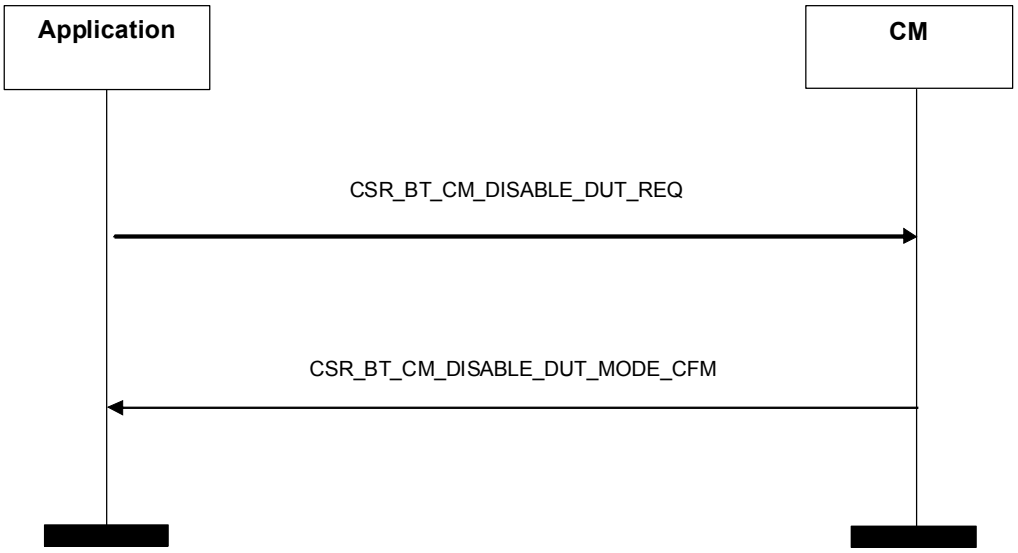


Figure 16: Disable device under test sequence

3.1.16 Service Search Procedure

The Service Search procedure is used for discovering the services being available on a specific Bluetooth device. For example, a laptop could search for a printer service when being used at a new location. The sequence, including both an initiator and a responder side, is outlined in Figure 17. The sequence starts when the application layer on the initiator side sends a CSR_BT_CM_SDC_SEARCH_REQ to the CM layer. The CM then opens a SDC channel and initiates a search for the service(s) on the peer device that corresponds to the one(s) requested.

An obtained service of the remote device is returned to the initiator in a CSR_BT_CM_SDC_SEARCH_IND signal. If the initiator has requested to search for more than one service, it will get a CSR_BT_CM_SDC_SEARCH_IND signal for each obtained service. E.g. if the initiator has requested to search for three services, but the peer device only has two of them, only two CSR_BT_CM_SDC_SEARCH_IND signals are returned. When the CM has finished searching for the service(s) it returns a CSR_BT_CM_SDC_SEARCH_CFM signal.

After receiving the CSR_BT_CM_SDC_SEARCH_CFM signal the initiator can either end the Service Search by sending a CSR_BT_CM_SDC_CLOSE_REQ, or request for an attribute value from one of the service records obtained in the CSR_BT_CM_SDC_SEARCH_IND signal(s) by sending a CSR_BT_CM_SDC_ATTRIBUTE_REQ signal.

When the initiator has obtained all the attribute values it needs, it must end the Service Search by sending a CSR_BT_CM_SDC_CLOSE_REQ. The Service Search procedure is finished when the initiator receives a CSR_BT_CM_SDC_CLOSE_IND signal.

Please note that if the CM cannot open a SDC channel or the peer device does not have any service that corresponds to the one(s) requested the initiator will receive a CSR_BT_CM_SDC_CLOSE_IND signal, meaning that the Service Search procedure is finished.

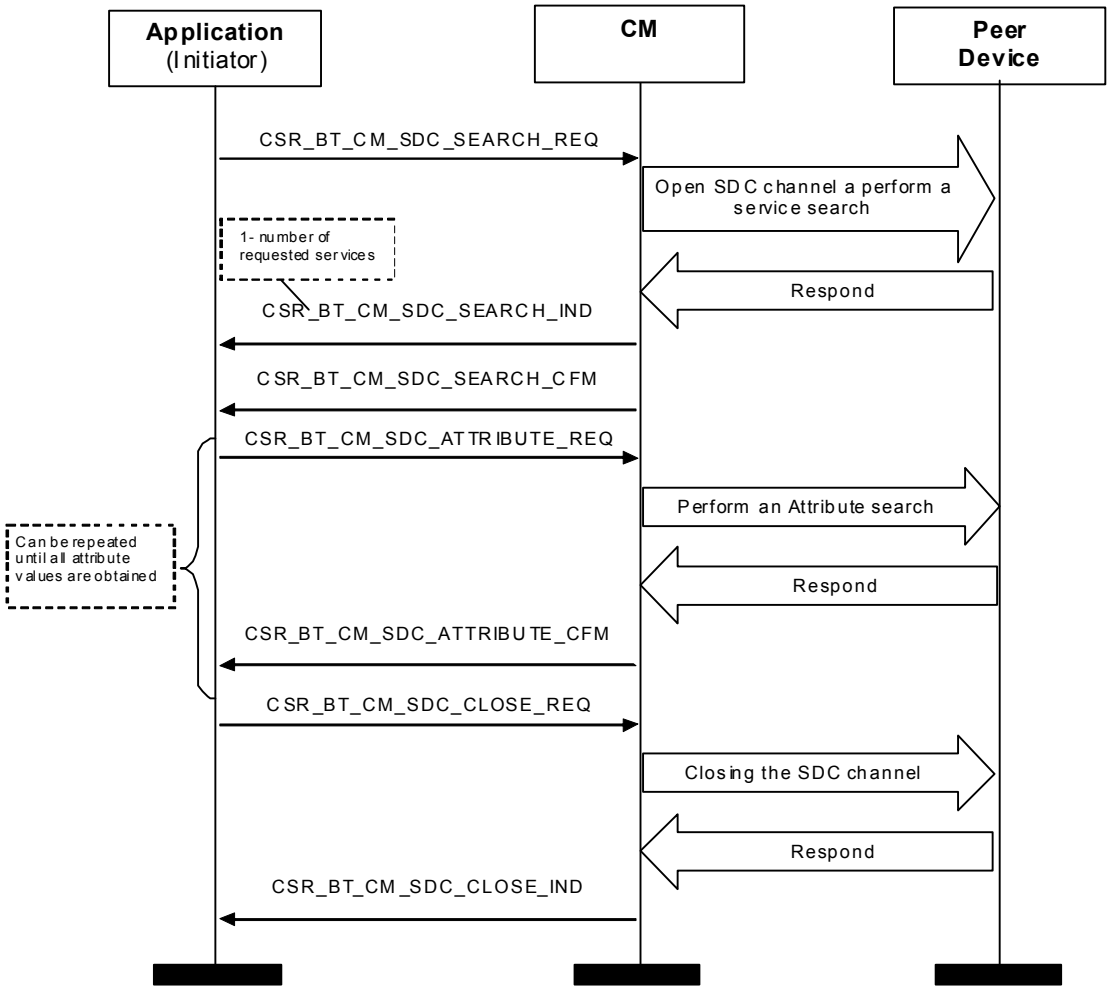


Figure 17: Service search sequence

3.1.17 Cancel Service Search Procedure

The application can cancel a CSR_BT_CM_SDC_SEARCH_REQ by sending a CSR_BT_CM_SDC_CANCEL_SEARCH_REQ. If the CSR_BT_CM_SDC_SEARCH_REQ is cancelled the application will received a CSR_BT_CM_SDC_CLOSE_IND.

Please notice that CSR_BT_CM_SDC_CANCEL_SEARCH_REQ is also used for cancelling a CSR_BT_CM_SDC_UUID128_SEARCH_REQ, see section 3.1.18, therefore the application **must** use the library function:

void CsrBtCmSdcCancelSearchReqSend (CsrSchedQid appHandle, deviceAddr_t deviceAddr)

to cancel a CSR_BT_CM_SDC_SEARCH_REQ. This function is defined in csr_bt_cm_lib.h.

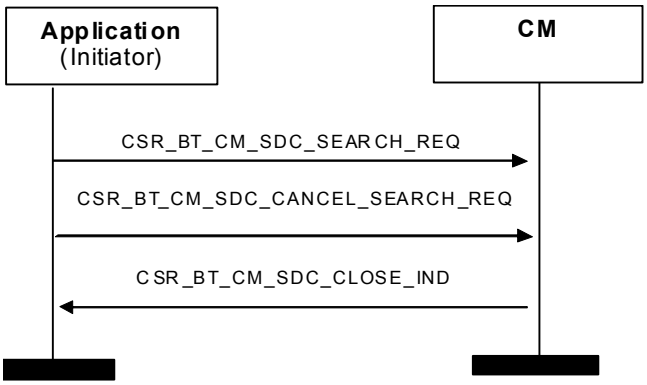


Figure 18: Cancel Service search sequence

3.1.18 UUID128 Service Search Procedure

This UUID128 Service Search procedure is related to the Service Search procedure in section 3.1.15. The UUID128 Service Search procedure is used if the application needs to do a searching for a service that is identified with a 128bit UUID, like SyncML. The sequence of the UUID128 Service Search procedure is outlined in Figure 19.

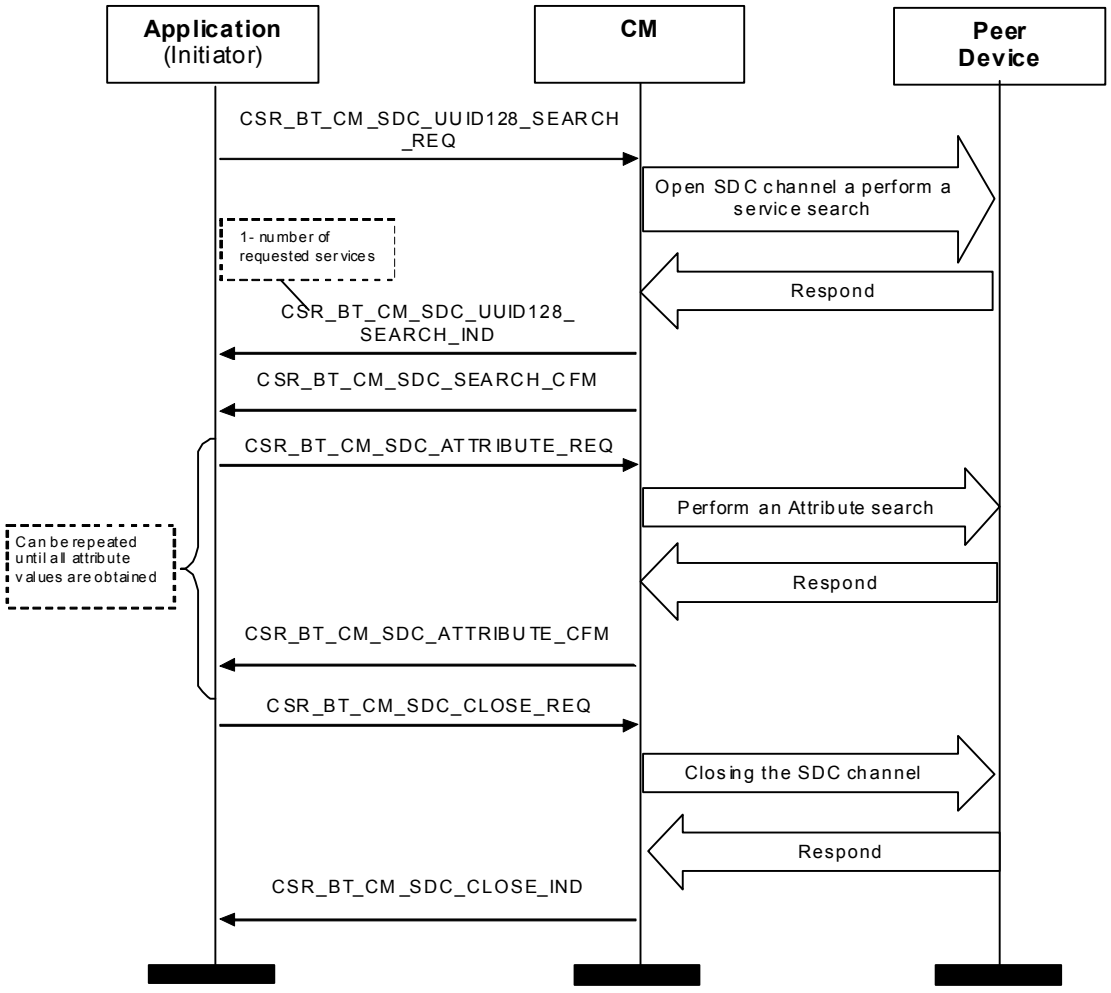


Figure 19: UUID128 Service search sequence

3.1.19 Cancel UUID128 Service Search Procedure

The application can cancel a CSR_BT_CM_SDC_UUID128_SEARCH_REQ by sending a CSR_BT_CM_SDC_CANCEL_SEARCH_REQ. If the CSR_BT_CM_SDC_UUID128_SEARCH_REQ is cancelled the application will receive a CSR_BT_CM_SDC_CLOSE_IND.

Please notice that CSR_BT_CM_SDC_CANCEL_SEARCH_REQ is also used for cancelling a CSR_BT_CM_SDC_SEARCH_REQ, see section 3.1.15, therefore the application **must** use the library function:

void CsrBtCmSdcCancelUuid128SearchReqSend (CsrSchedQid appHandle, deviceAddr_t deviceAddr)

to cancel a CSR_BT_CM_SDC_UUID128_SEARCH_REQ. This function is defined in csr_bt_cm_lib.h.

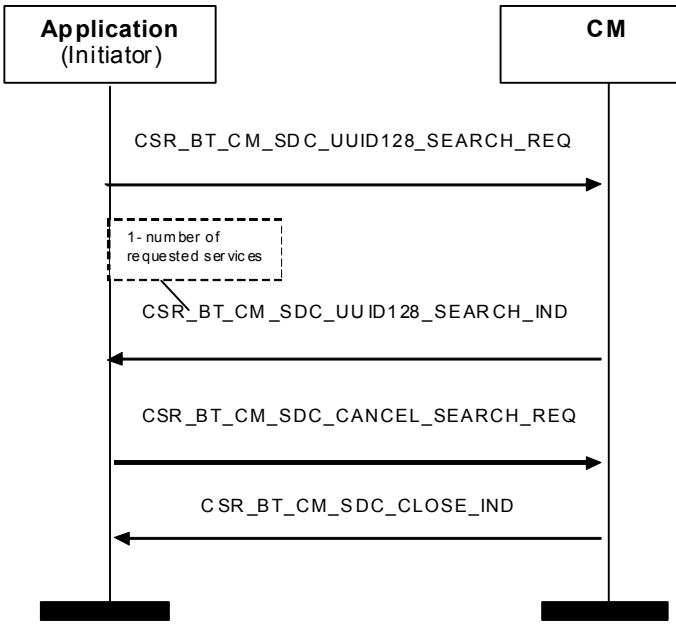


Figure 20: Cancel UUID128Service search sequence

3.1.20 Read Remote Extended Features

The remote extended features of a device can be requested.

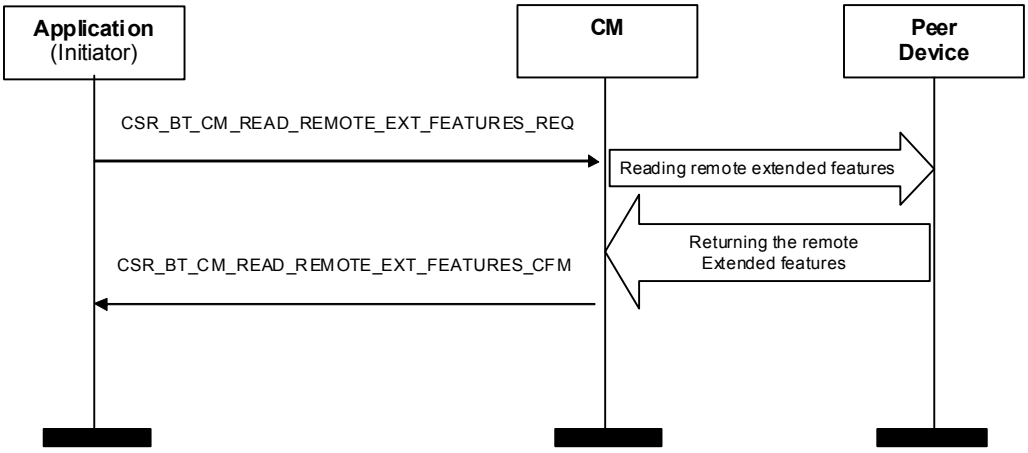


Figure 21: Read Remote Extended Features

Please notice that if the remote device is older than Bluetooth version 1.2 the connection manager will automatically attempt to retrieve the remote features using non-extended requests. For more details refer to 4.21.

3.1.21 Set AFH Channel Classification

The AFH channel classification can be written to the device.

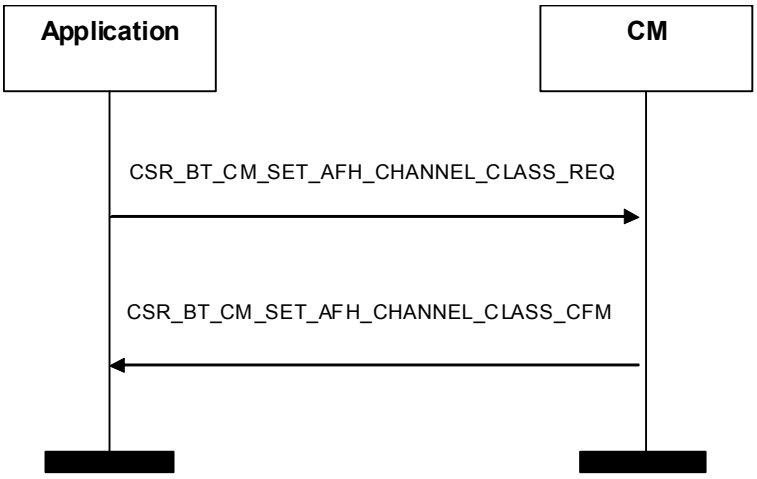


Figure 22: Set AFH Channel Classification

Please notice that this message is only valid if the local device supports Bluetooth version 1.2 or higher. In case the local device does not support Bluetooth version 1.2 or higher an error result is returned. For more details refer to 0.

3.1.22 Read AFH Channel Assessment Mode

Read the status of the channel assessment mode.

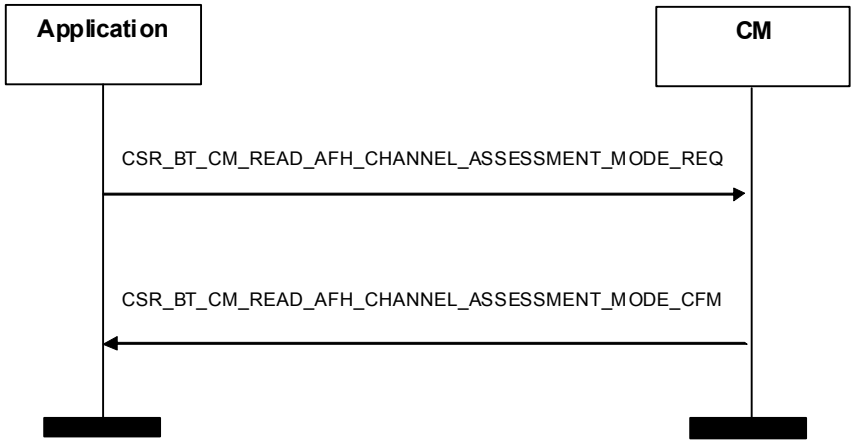


Figure 23: Read AFH Channel Assessment Mode

Please notice that this message is only valid if the local device supports Bluetooth version 1.2 or higher. In case the local device does not support Bluetooth version 1.2 or higher an error result is returned. For more details refer to 4.23.

3.1.23 Write AFH Channel Assessment Mode

Change the status of the channel assessment mode.

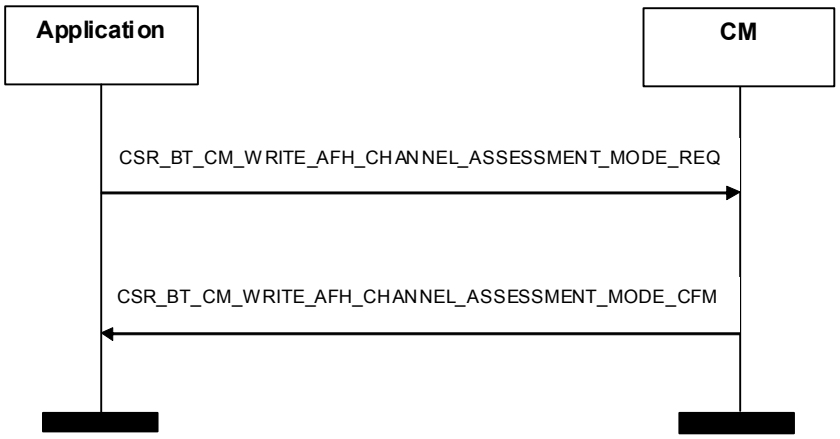


Figure 24: Write AFH Channel Assessment Mode

Please notice that this message is only valid if the local device supports Bluetooth version 1.2 or higher. In case the local device does not support Bluetooth version 1.2 or higher an error result is returned. For more details refer to 4.24.

3.1.24 Read Local Extended Features

Read the local extended features of a device.

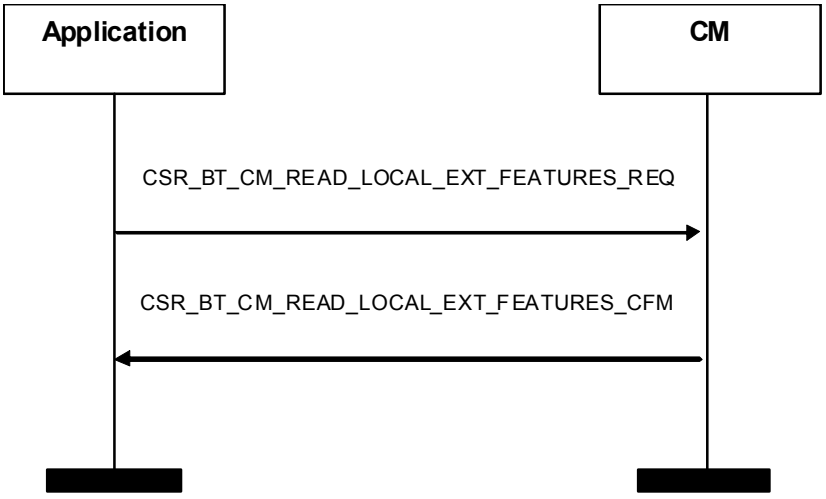


Figure 25: Read Local Extended Features

Please notice that this message is only valid if the local device supports Bluetooth version 1.2 or higher. In case the local device does not support Bluetooth version 1.2 or higher an error result is returned. For more details refer to 4.25.

3.1.25 Read AFH Channel Map

Read the AFH channel classification map and the status of the AFH mode.

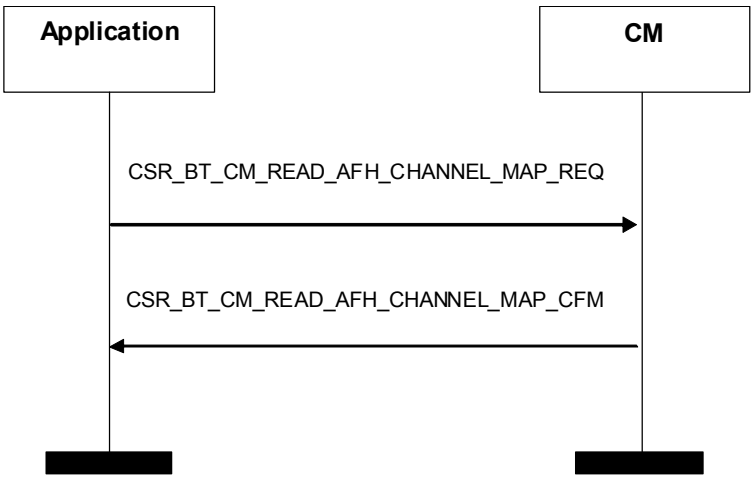


Figure 26: Read AFH Channel Map

Please notice that this message is only valid if the local and the remote device support Bluetooth version 1.2 or higher. If one of the devices does not support Bluetooth version 1.2 or higher an error result is returned. For more details refer to 4.26.

3.1.26 Read Clock

Read the local clock or the piconet clock.

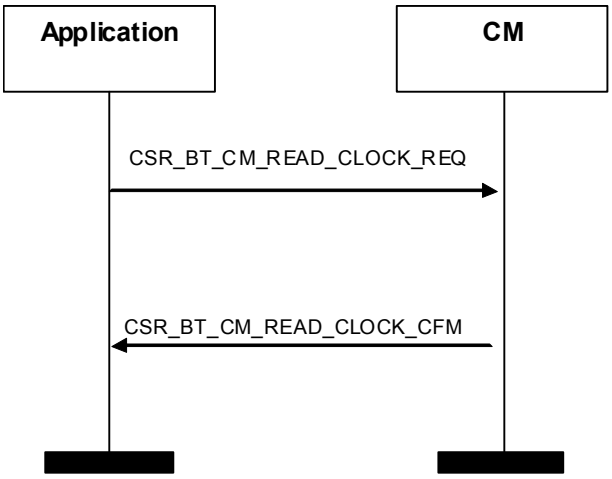


Figure 27: Read Clock

Please notice that this message is only valid if the local and the remote device support Bluetooth version 1.2 or higher. If one of the devices does not support Bluetooth version 1.2 or higher an error result is returned. For more details refer to 4.27.

3.1.27 Read TX Power Level

This command will read the values of the transmit power level parameter for a specified connection.

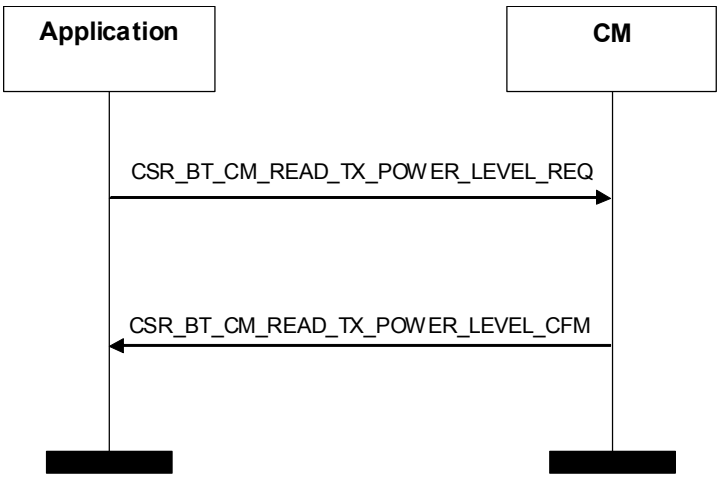


Figure 28: Read TX power level

3.1.28 Get Link Quality

This command will return the value of the link quality for a specified connection. This command will return a link quality value from 0-255 which represents the quality of the link between two Bluetooth devices. The higher the value, the better the link quality is.

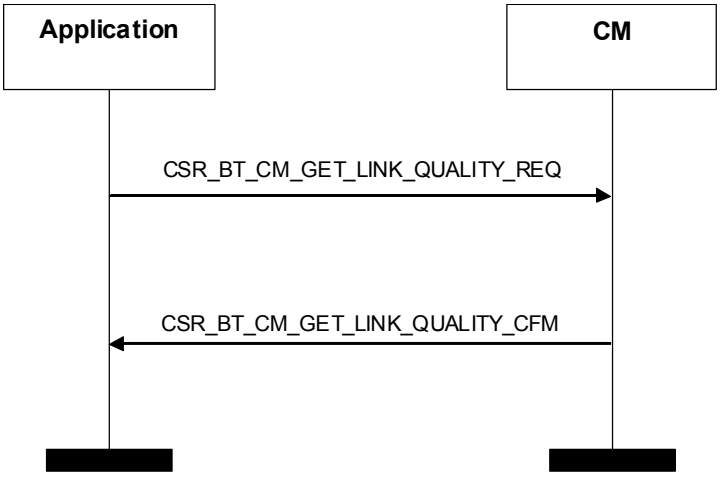


Figure 29: Get link quality

3.1.29 Read RSSI

This command will read the value for the difference between the measured Received Signal Strength Indication (RSSI) and the limit of the Golden Received Power Range of a specified connection.

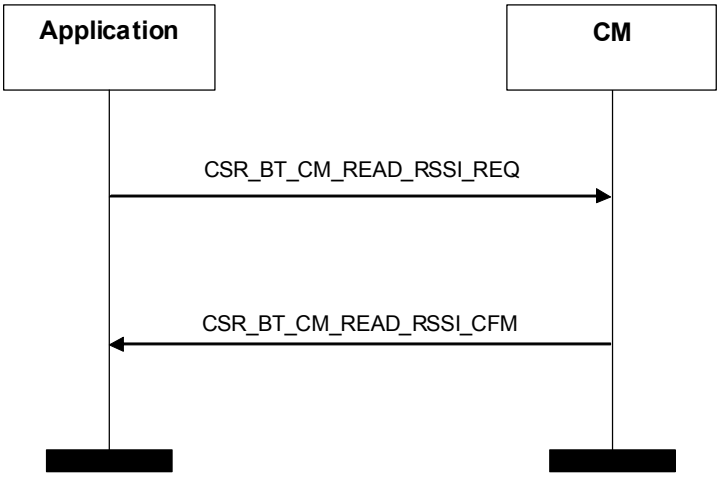


Figure 30: Read RSSI

3.1.30 Read Local Name

The local device name can be read by sending a CSR_BT_CM_READ_LOCAL_NAME_REQ.

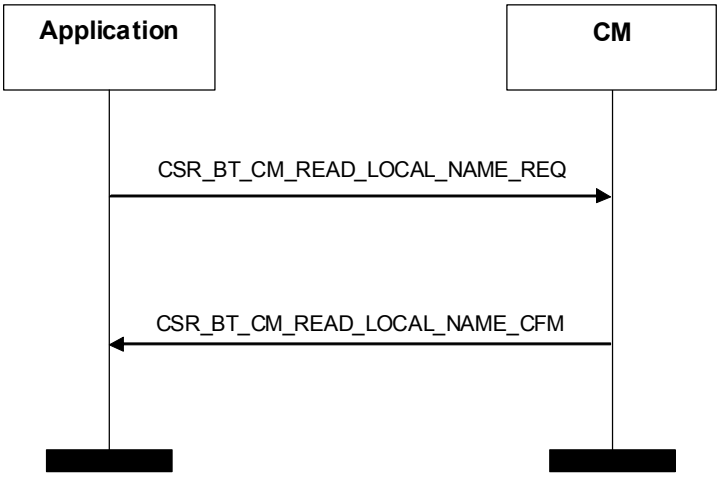


Figure 31: Read local name

3.1.31 Write Page To

This command will write the value of the page timeout window. The page timeout window defines the maximum time CSR Synergy Bluetooth will wait for a baseband page response from a remote device at a locally initiated connection attempt.

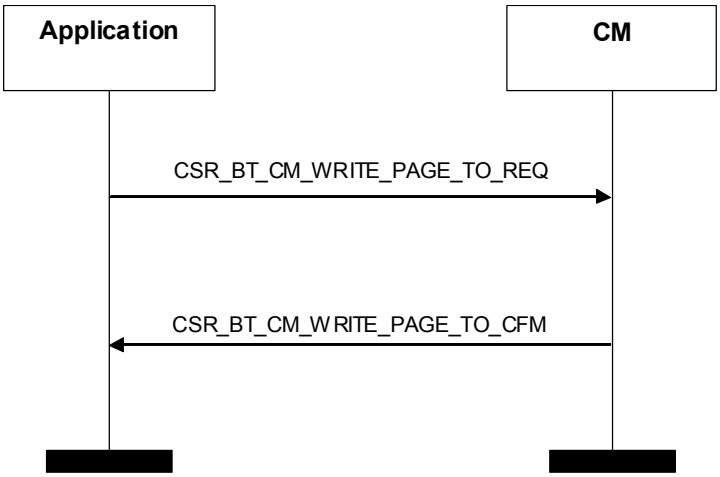


Figure 32: Write page to

Please notice that default value of the page timeout window is defined in csr_bt_usr_config.h as PAGE_TIMEOUT.

3.1.32 Write Link Policy

This command can be used for controlling which policy settings - i.e. sniff/park are allowed .

Figure 33: Write Link Policy

3.1.33 Read Link Policy

This command can be used for reading which policy settings - i.e. sniff/park are allowed

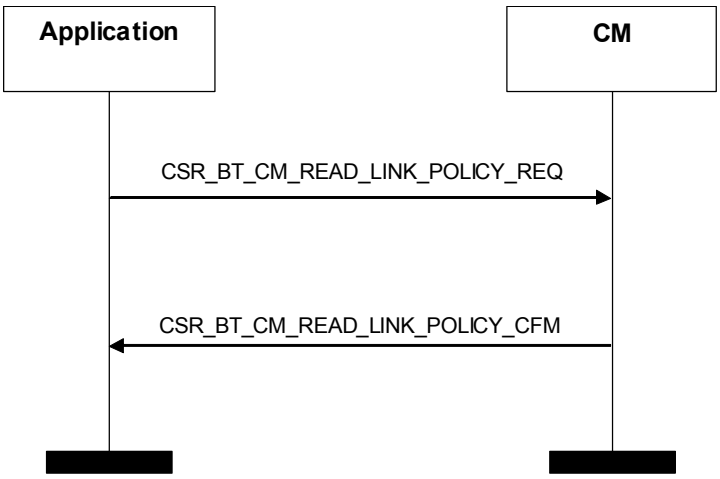


Figure 34: Read Link Policy

3.1.34 Write Class of Device

This command can be used for controlling the class of device (both service and major/minor class of device).

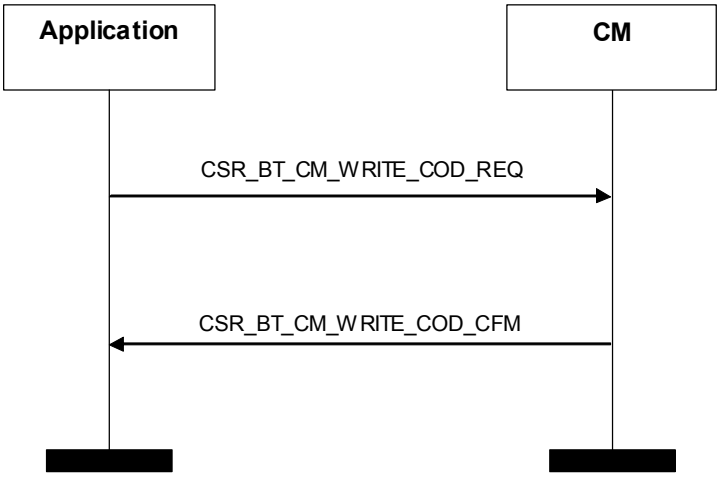


Figure 35: Write Class of Device

3.1.35 Read Class of Device

This command can be used for reading the Class of Device settings

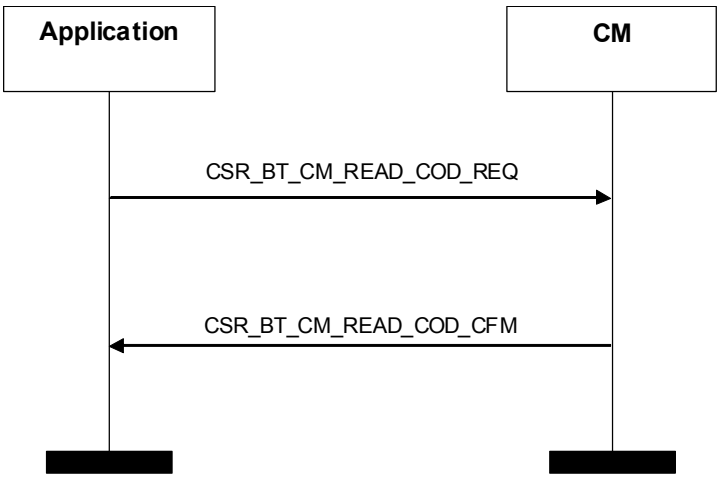


Figure 36: Read Class of Device

3.1.36 Read Local Version

Read the local version of a device.

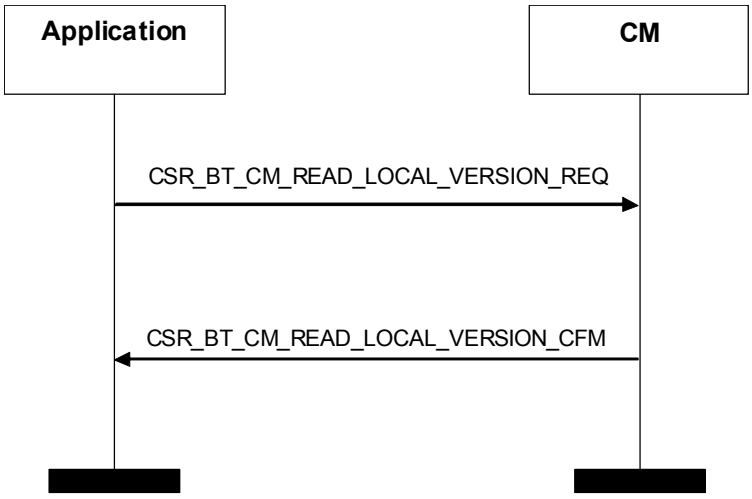


Figure 37: Read Local Extended Features

3.1.37 Role Switch Config

This command can be used for controlling role switch behaviour, e.g. when the CM will try and force a role switch with the remote device.

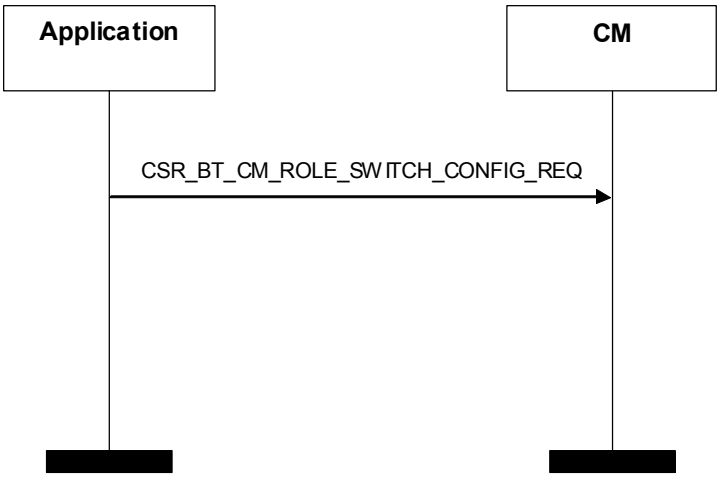


Figure 38: Role Switch Config

3.1.38 ACL Detach Request

This signal can be used for detaching an already established ACL.

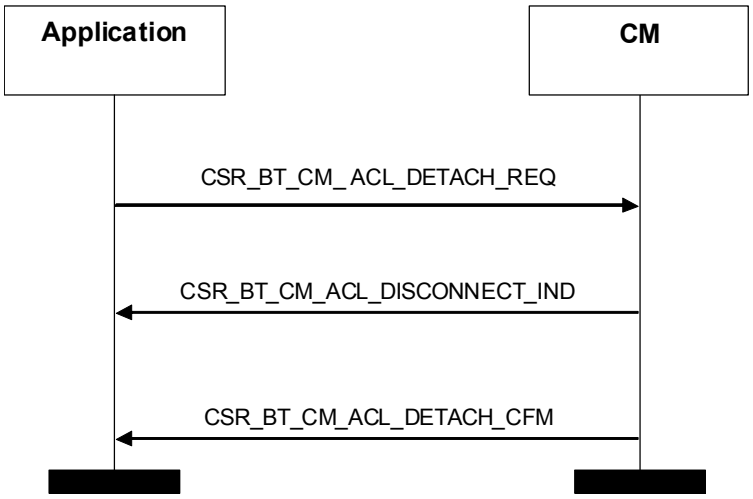


Figure 39: ACL Detach

Note, the CSR_BT_CM_ACL_DISCONNECT_IND is only sent if the application has subscribed for this event, see section 4.43.

3.1.39 Read Failed Contact Counter

This command can be used for reading the controllers failed contact counter for a particular device.

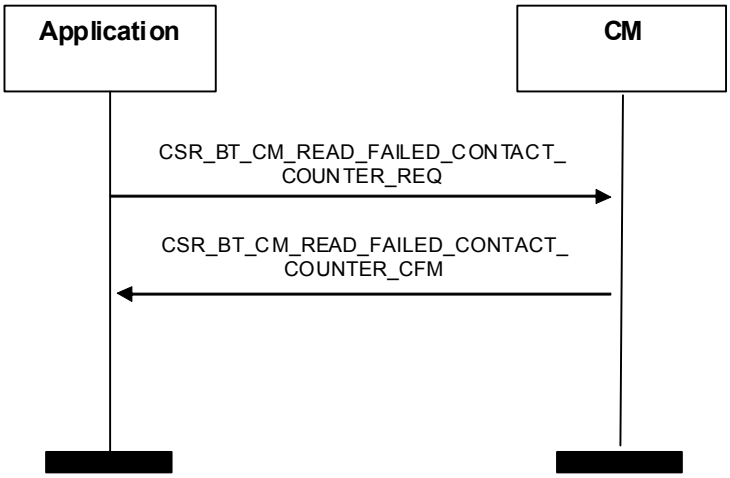


Figure 40: Read Failed Contact Counter

3.1.40 Set Event Mask

This signal can be used for setting which extended information the application will subscribe for, and may be sent momentarily from the application. In the request message the application can defined which extended information it will subscribe for, and the confirm message defines which event has been set. The extended information the application can subscribe for is defined in section 4.43.

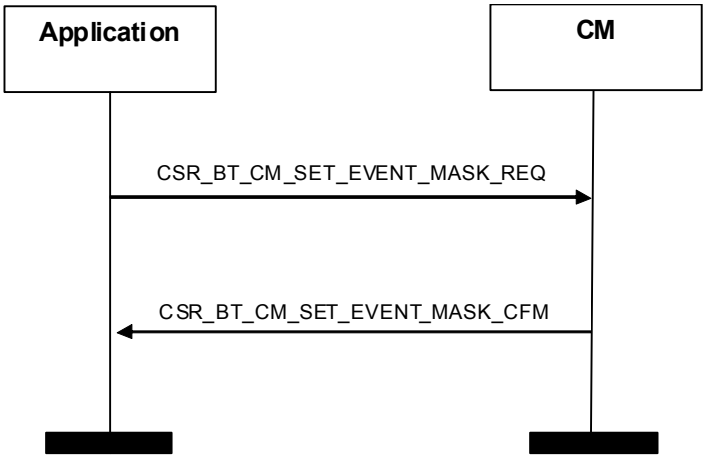


Figure 41: Set Event Mask

3.1.41 Mode Change Config

This command can be used for configuring if CSR Synergy Bluetooth or the application must control the link power behavior.

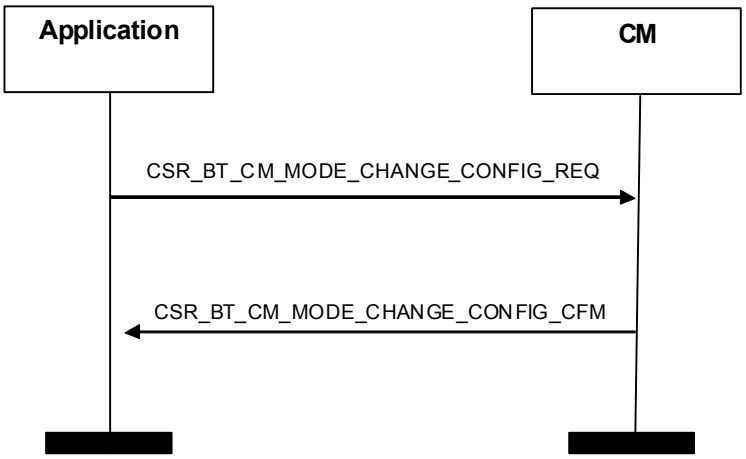


Figure 42: Mode Change Config

3.1.42 Mode Change

This command can be used for either trying to set a given ACL link in Sniff Mode or exiting Sniff mode.

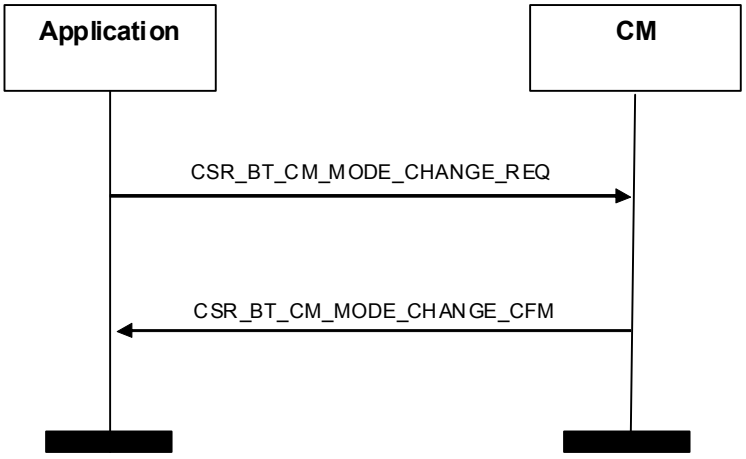


Figure 43: Mode Change

3.1.43 Register

This command may be used by applications using the RFCOMM protocol in order to allocate a specific server channel.

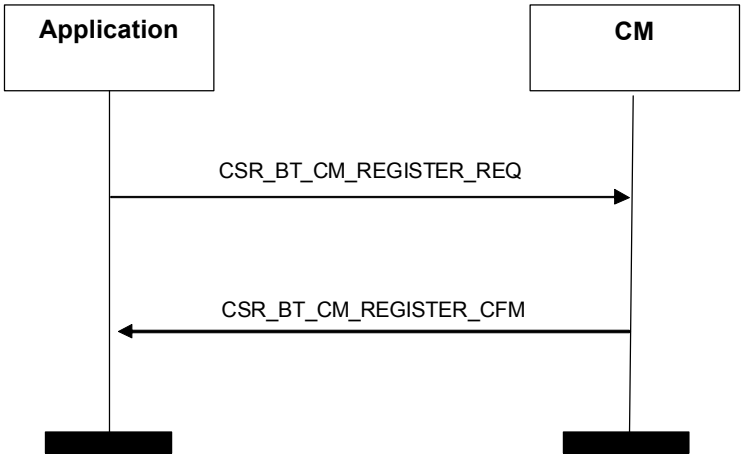


Figure 44: Register

3.1.44 Unregister

This command may be used by an application that has allocated a specific server channel in order to free it again.

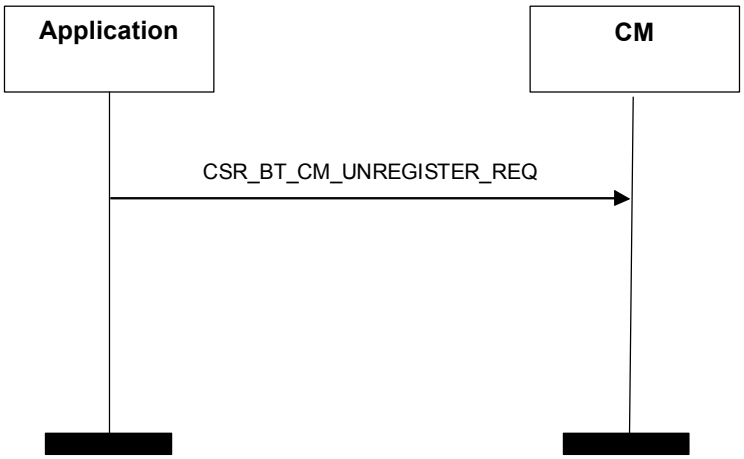


Figure 45: Register

3.1.45 Read Advertising Channel TX Power

This command will read the advertising channel TX power level.

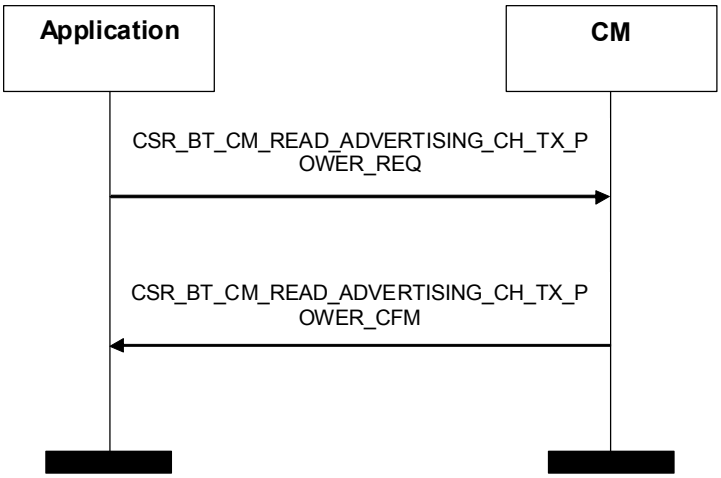


Figure 46: Read Advertising Channel TX Power Level

4 Connection Manager Primitives

This section gives an overview of the primitives and parameters in the interface. Detailed information can be found in the corresponding csr_bt_cm_prim.h file.

4.1 List of All Primitives

Primitives	Reference
CSR_BT_CM_SET_LOCAL_NAME_REQ	See section 4.2
CSR_BT_CM_SET_LOCAL_NAME_CFM	See section 4.2
CSR_BT_CM_READ_LOCAL_BD_ADDR _REQ	See section 4.3
CSR_BT_CM_READ_LOCAL_BD_ADDR _CFM	See section 4.3
CSR_BT_CM_WRITE_LINK_SUPERV_TIMEOUT_REQ	See section 4.4
CSR_BT_CM_WRITE_LINK_SUPERV_TIMEOUT_CFM	See section 4.4
CSR_BT_CM_READ_REMOTE_NAME_REQ	See section 4.5
CSR_BT_CM_READ_REMOTE_NAME_CFM	See section 4.5
CSR_BT_CM_CANCEL_READ_REMOTE_NAME_REQ	See section 4.6
CSR_BT_CM_READ_REMOTE_VERSION_REQ	See section 4.7
CSR_BT_CM_READ_REMOTE_VERSION_CFM	See section 4.7
CSR_BT_CM_WRITE_SCAN_ENABLE_REQ	See section 4.7
CSR_BT_CM_WRITE_SCAN_ENABLE_CFM	See section 4.7
CSR_BT_CM_READ_SCAN_ENABLE_REQ	See section 4.9
CSR_BT_CM_READ_SCAN_ENABLE_CFM	See section 4.9
CSR_BT_CM_WRITE_PAGESCAN_SETTINGS_REQ	See section 4.10
CSR_BT_CM_WRITE_PAGESCAN_SETTINGS_CFM	See section 4.10
CSR_BT_CM_WRITE_PAGESCAN_TYPE_REQ	See section 4.11
CSR_BT_CM_WRITE_PAGESCAN_TYPE_CFM	See section 4.11
CSR_BT_CM_WRITE_INQUIRYSCAN_SETTINGS_REQ	See section 4.12
CSR_BT_CM_WRITE_INQUIRYSCAN_SETTINGS_CFM	See section 4.12
CSR_BT_CM_WRITE_INQUIRYSCAN_TYPE_REQ	See section 4.13
CSR_BT_CM_WRITE_INQUIRYSCAN_TYPE_CFM	See section 4.13
CSR_BT_CM_CONNECTABLE_REQ	See section 4.14
CSR_BT_CM_REJECT_RFC_CONNECTION_IND	See section 4.15
CSR_BT_CM_ENABLE_DUT_MODE_REQ	See section 4.16
CSR_BT_CM_ENABLE_DUT_MODE_CFM	See section 4.16
CSR_BT_CM_DISABLE_DUT_MODE_REQ	See section 4.17
CSR_BT_CM_DISABLE_DUT_MODE_CFM	See section 4.17
CSR_BT_CM_SDC_SEARCH_REQ	See section 4.17
CSR_BT_CM_SDC_SEARCH_IND	See section 4.17
CSR_BT_CM_SDC_SEARCH_CFM	See section 4.17
CSR_BT_CM_SDC_ATTRIBUTE_REQ	See section 4.17
CSR_BT_CM_SDC_ATTRIBUTE_CFM	See section 4.17
CSR_BT_CM_SDC_CLOSE_REQ	See section 4.20
CSR_BT_CM_SDC_CLOSE_IND	See section 4.20

Primitives	Reference
CSR_BT_CM_READ_REMOTE_EXT_FEATURES_REQ	See section 4.21
CSR_BT_CM_READ_REMOTE_EXT_FEATURES_CFM	See section 4.21
CSR_BT_CM_SET_AFH_CHANNEL_CLASS_REQ	See section 4.22
CSR_BT_CM_SET_AFH_CHANNEL_CLASS_CFM	See section 4.22
CSR_BT_CM_READ_AFH_CHANNEL_ASSESSMENT_MODE_REQ	See section 4.23
CSR_BT_CM_READ_AFH_CHANNEL_ASSESSMENT_MODE_CFM	See section 4.23
CSR_BT_CM_WRITE_AFH_CHANNEL_ASSESSMENT_MODE_REQ	See section 4.24
CSR_BT_CM_WRITE_AFH_CHANNEL_ASSESSMENT_MODE_CFM	See section 4.24
CSR_BT_CM_READ_LOCAL_EXT_FEATURES_REQ	See section 4.25
CSR_BT_CM_READ_LOCAL_EXT_FEATURES_CFM	See section 4.25
CSR_BT_CM_READ_AFH_CHANNEL_MAP_REQ	See section 4.26
CSR_BT_CM_READ_AFH_CHANNEL_MAP_CFM	See section 4.26
CSR_BT_CM_READ_CLOCK_REQ	See section 4.27
CSR_BT_CM_READ_CLOCK_CFM	See section 4.27
CSR_BT_CM_READ_TX_POWER_LEVEL_REQ	See section 4.28
CSR_BT_CM_READ_TX_POWER_LEVEL_CFM	See section 4.28
CSR_BT_CM_GET_LINK_QUALITY_REQ	See section 4.29
CSR_BT_CM_GET_LINK_QUALITY_CFM	See section 4.29
CSR_BT_CM_READ_RSSI_REQ	See section 4.30
CSR_BT_CM_READ_RSSI_CFM	See section 4.30
CSR_BT_CM_READ_LOCAL_NAME_REQ	See section 4.31
CSR_BT_CM_READ_LOCAL_NAME_CFM	See section 4.31
CSR_BT_CM_WRITE_PAGE_TO_REQ	See section 4.32
CSR_BT_CM_WRITE_PAGE_TO_CFM	See section 4.32
CSR_BT_CM_SDC_UUID128_SEARCH_REQ	See section 4.33
CSR_BT_CM_SDC_UUID128_SEARCH_IND	See section 4.33
CSR_BT_CM_SDC_CANCEL_SEARCH_ REQ	See section 4.34
CSR_BT_CM_ROLE_DISCOVERY_REQ	See section 4.35
CSR_BT_CM_ROLE_DISCOVERY_CFM	See section 4.35
CSR_BT_CM_WRITE_LINK_POLICY_REQ	See section 4.36
CSR_BT_CM_WRITE_LINK_POLICY_ERROR_IND	See section 4.36
CSR_BT_CM_READ_LINK_POLICY_REQ	See section 4.37
CSR_BT_CM_READ_LINK_POLICY_CFM	See section 4.37
CSR_BT_CM_EIR_UPDATE_MANUFACTURER_DATA_REQ	See section 4.38
CSR_BT_CM_EIR_UPDATE_MANUFACTURER_DATA_CFM	See section 4.38
CSR_BT_CM_WRITE_COD_REQ	See section 4.39
CSR_BT_CM_WRITE_COD_CFM	See section 4.39
CSR_BT_CM_READ_COD_REQ	See section 4.40
CSR_BT_CM_READ_COD_CFM	See section 4.40
CSR_BT_CM_READ_LOCAL_VERSION_REQ	See section 4.41
CSR_BT_CM_READ_LOCAL_VERSION_CFM	See section 4.41
CSR_BT_CM_ROLE_SWITCH_CONFIG_REQ	See section 4.42

Primitives	Reference
CSR_BT_CM_SET_EVENT_MASK_REQ	See section 4.43
CSR_BT_CM_SET_EVENT_MASK_CFM	See section 4.43
CSR_BT_CM_SYNC_CONNECT_IND	See section 4.44
CSR_BT_CM_SYNC_RENEGOTIATE_IND	See section 4.44
CSR_BT_CM_SYNC_DISCONNECT_IND	See section 4.44
CSR_BT_CM_EXT_SYNC_CONNECT_IND	See section 4.44
CSR_BT_CM_MODE_CHANGE_IND	See section 4.45
CSR_BT_CM_SNIFF_SUB-RATING_IND	See section 4.45
CSR_BT_CM_ROLE_CHANGE_IND	See section 4.46
CSR_BT_CM_LSTO_CHANGE-IND	See section 4.47
CSR_BT_CM_BLUECORE_INITIALIZED_IND	See section 4.48
CSR_BT_CM_ACL_DISCONNECT_IND	See section 4.49
CSR_BT_CM_ACL_CONNECT_IND	See section 4.49
CSR_BT_CM_ACL_DETACH_REQ	See section 4.50
CSR_BT_CM_ACL_DETACH_CFM	See section 4.50
CSR_BT_CM_READ_FAILED_CONTACT_COUNTER_REQ	See section 4.51
CSR_BT_CM_READ_FAILED_CONTACT_COUNTER_CFM	See section 4.51
CSR_BT_CM_SWITCH_ROLE_REQ	See section 4.52
CSR_BT_CM_SWITCH_ROLE _CFM	See section 4.52
CSR_BT_CM_MODE_CHANGE_CONFIG_REQ	See section 4.53
CSR_BT_CM_MODE_CHANGE_CONFIG_CFM	See section 4.53
CSR_BT_CM_MODE_CHANGE_REQ	See section 4.54
CSR_BT_CM_MODE_CHANGE_CFM	See section 4.54
CSR_BT_CM_LOGICAL_CHANNEL_TYPE_IND	See section 4.55
CSR_BT_CM_READ_REMOTE_FEATURES_IND	See section 4.56
CSR_BT_CM_A2DP_BIT_RATE_IND	See section 4.57
CSR_BT_CM_INQUIRY_PAGE_EVENT_IND	See section 4.58
CSR_BT_CM_BLE_CONNECTION_IND	See section 4.59
CSR_BT_CM_ENCRYPT_CHANGE_IND	See section 4.60
CSR_BT_CM_ALWAYS_MASTER_DEVICES_REQ	See section 4.61
CSR_BT_CM_ALWAYS_MASTER_DEVICES_CFM	See section 4.61
CSR_BT_CM_REGISTER_REQ	See section 4.62
CSR_BT_CM_REGISTER_CFM	See section 4.62
CSR_BT_CM_UNREGISTER_REQ	See section 4.63
CSR_BT_CM_READ_ADVERTISING_CH_TX_POWER_REQ	See section 4.64
CSR_BT_CM_READ_ADVERTISING_CH_TX_POWER_CFM	See section 4.64
CSR_BT_CM_LOCAL_NAME_CHANGE_IND	See section 4.65
CSR_BT_CM_LE_EVENT_ADVERTISING_IND	See section 4.66
CSR_BT_CM_LE_EVENT_SCAN_IND	See section 4.67
CSR_BT_CM_LE_EVENT_CONNECTION_IND	See section 4.68

Table 1: List of all primitives

4.2 CSR_BT_CM_SET_LOCAL_NAME

Primitives \ Parameters	type	phandle	friendlyName	resultCode	resultSupplier
CSR_BT_CM_SET_LOCAL_NAME_REQ	✓	✓	✓		
CSR_BT_CM_SET_LOCAL_NAME_CFM	✓			✓	✓

Table 2: CSR_BT_CM_SET_LOCAL_NAME Primitives

Description

Change the Bluetooth® device name in the local device. This name can be retrieved by other devices, e.g. during device discovery.

Parameters

type	Signal identity, CSR_BT_CM_SET_LOCAL_NAME_REQ/CFM.
phandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.
friendlyName	Pointer to zero-terminated utf8 string containing the name of the local device.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.3 CSR_BT_CM_READ_LOCAL_BD_ADDR

Primitives \ Parameters	type	phandle	deviceAddr
CSR_BT_CM_READ_LOCAL_BD_ADDR_REQ	✓	✓	
CSR_BT_CM_READ_LOCAL_BD_ADDR_CFM	✓		✓

Table 3: CSR_BT_CM_READ_LOCAL_BD_ADDR Primitives

Description

Read the local Bluetooth® address.

Parameters

type	Signal identity, CSR_BT_CM_READ_LOCAL_BD_ADDR _REQ/CFM.
phandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.
deviceAddr	The Bluetooth® address of the local device.

4.4 CSR_BT_CM_WRITE_LINK_SUPERV_TIMEOUT

Primitives \ Parameters	type	phandle	deviceAddr	resultCode	resultSupplier	timeout
CSR_BT_CM_WRITE_LINK_SUPERV_TIMEOUT_REQ	✓	✓	✓			✓
CSR_BT_CM_WRITE_LINK_SUPERV_TIMEOUT_CFM	✓		✓	✓	✓	

Table 4: CSR_BT_CM_WRITE_LINK_SUPERV_TIMEOUT Primitives

Description

This command will write the value for the Link Supervision Timeout parameter for the given device address. The Link Supervision timeout parameter is used by the master or slave Bluetooth® device to monitor link loss. If, for any reason, no Baseband packets are received from this Bluetooth® device for a duration longer than the timeout value, the connection is released. The same timeout value is used for both SCO and ACL connections.

The Link Supervision timeout value is measured in Number of Baseband slots, e.g. Link Supervision timeout value = N * 0,625ms where range for N is 0x0001 – 0xFFFF. Please note that a Link Supervision timeout value of 0x0000 will disable the Link Supervision timeout check. The default value for the Link Supervision Timeout on newly created ALC links is 0x7D00, which is 20 seconds. This default value can be adjusted by changing the DEFAULT_LINK_SUPERVISION_TIMEOUT value in usr_config.h.

Parameters

type	Signal identity, CSR_BT_CM_WRITE_LINK_SUPERV_TIMEOUT_REQ/CFM.
phandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.
deviceAddr	The Bluetooth® address of the link which the supervision timeout value.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
timeout	The Link Supervision timeout value.

4.5 CSR_BT_CM_READ_REMOTE_NAME

Primitives \ Parameters	type	phandle	deviceAddr	resultCode	resultSupplier	friedlyname
CSR_BT_CM_READ_REMOTE_NAME_REQ	✓	✓	✓			
CSR_BT_CM_READ_REMOTE_NAME_CFM	✓		✓	✓	✓	✓

Table 5: CSR_BT_CM_READ_REMOTE_NAME Primitives

Description

This command will request the remote (friendly) name of a remote device, which is identified by its Bluetooth Device Address.

Note: Issuing a CSR_BT_CM_READ_REMOTE_NAME_REQ might cause existing ACL links to be dropped. This can occur on ACL links where the local device is slave. To circumvent this, the application may use CSR_BT_CM_ROLE_SWITCH_CONFIG_REQ (see section 4.44) by specifying CSR_BT_CM_ROLE_SWITCH_BEFORE_RNR. This will instruct the Connection Manager to perform role switch on links where the local device is slave. This will reduce the likelihood of existing links dropping due to link supervision timeout.

Parameters

type	Signal identity, CSR_BT_CM_READ_REMOTE_NAME_REQ/CFM.
phandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.
deviceAddr	The Bluetooth® address of the remote device
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
friendlyName	Pointer to zero-terminated utf-8 string containing the name of the remote device.

4.6 CSR_BT_CM_CANCEL_READ_REMOTE_NAME

Primitives \ Parameters	type	appHandle	deviceAddr
CSR_BT_CM_CANCEL_READ_REMOTE_NAME_REQ	✓	✓	✓

Table 6: CSR_BT_CM_CANCEL_READ_REMOTE_NAME Primitives

Description

This command will cancel a previous CSR_BT_CM_READ_REMOTE_NAME_REQ identified by Bluetooth® address and appHandle. If there are no pending or ongoing procedures for the specified appHandle/deviceAddr combination, the CSR_BT_CM_CANCEL_READ_REMOTE_NAME_REQ is ignored.

Parameters

type	Signal identity, CSR_BT_CM_READ_REMOTE_NAME_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
deviceAddr	The Bluetooth® address of the remote device

4.7 CSR_BT_CM_READ_REMOTE_VERSION

Primitives \ Parameters	type	appHandle	deviceAddr	resultCode	resultSupplier	impVersion	manufacturerName	lmpSubversion
CSR_BT_CM_READ_REMOTE_VERSION_REQ	✓	✓	✓					
CSR_BT_CM_READ_REMOTE_VERSION_CFM	✓		✓	✓	✓	✓	✓	✓
CSR_BT_CM_READ_REMOTE_VERSION_IND	✓		✓	✓	✓	✓	✓	✓

Table 7: CSR_BT_CM_READ_REMOTE_VERSION Primitives

Description

This command will request to read the version of a remote device, which is identified by its Bluetooth Device Address. If the application has subscribed for Read Remote Version Event by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_REMOTE_VERSION, it will receive the CSR_BT_CM_READ_REMOTE_VERSION_IND message every time the Remote Version has been read.

Parameters

type	Signal identity, CSR_BT_CM_READ_REMOTE_VERSION_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
deviceAddr	The Bluetooth® address of the remote device
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
lmpVersion	Version of the Current LMP in the remote Bluetooth device. For LMP_Version information see: https://www.bluetooth.org/foundry/assignnumb/document/link_manager_protocol
manufacturerName	Manufacturer Name of the remote Bluetooth device see: https://www.bluetooth.org/foundry/assignnumb/document/company_identifiers
lmpSubversion	Subversion of the Current LMP in the remote Bluetooth device which is defined by each company.

4.8 CSR_BT_CM_WRITE_SCAN_ENABLE

Parameters / Primitives	type	appHandle	disableInquiryScan	disablePageScan	resultCode	resultSupplier
CSR_BT_CM_WRITE_SCAN_ENABLE_REQ	✓	✓	✓	✓		
CSR_BT_CM_WRITE_SCAN_ENABLE_CFM	✓				✓	✓

Table 8: CSR_BT_CM_WRITE_SCAN_ENABLE Primitives

Description

The Write Scan Enable command controls whether or not the local Bluetooth device will periodically scan for page attempts and/or inquiry requests from other Bluetooth® devices.

Please note that if the parameter *disablePageScan* is set to FALSE the local device will not scan for page requests from other devices before a profile is activated. However if *disablePageScan* is set to TRUE and a profile is activated it will never scan for page scan requests.

Please notice that under initialization of CSR Synergy Bluetooth the parameters *disableInquiryScan* and *disablePageScan* are set to FALSE, e.g. the local device is setup to scan periodically for inquiry requests from other Bluetooth® devices and will scan for page scan requests when a profile is activated.

Parameters

type	Signal identity, CSR_BT_CM_WRITE_SCAN_ENABLE_REQ/CFM.
appHandle	The identity of the calling process.
disableInquiryScan	If disableInquiryScan is set to TRUE, the local Bluetooth® device will not scan for inquiry requests from other Bluetooth® devices.
disablePageScan	If disablePageScan is set to TRUE, the local Bluetooth® device will not scan for page requests from other Bluetooth® devices even if another profile is activate.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.9 CSR_BT_CM_READ_SCAN_ENABLE

Primitives \ Parameters	type	appHandle	resultCode	resultSupplier	scanEnable
CSR_BT_CM_READ_SCAN_ENABLE_REQ	✓	✓			
CSR_BT_CM_READ_SCAN_ENABLE_CFM	✓		✓	✓	✓

Table 9: CSR_BT_CM_READ_SCAN_ENABLE Primitives

Description

The Read Scan Enable command will read the value for the Scan Enable configuration parameter, which controls whether or not the Bluetooth device will periodically scan for page attempts and/or inquiry requests from other Bluetooth devices.

Parameters

type — Signal identity, CSR_BT_CM_READ_SCAN_ENABLE _REQ/CFM.

appHandle — The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.

resultCode — The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.

resultSupplier — This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

scanEnable — scanEnable can have the following values, which are defined in csr_bt_profiles.h

HCI_SCAN_ENABLE_OFF, meaning that No Scans is enabled.

HCI_SCAN_ENABLE_INQ, meaning that Inquiry Scan is enabled and Page Scan is disabled.

HCI_SCAN_ENABLE_PAGE, meaning that Inquiry Scan is disabled and Page Scan is enabled.

HCI_SCAN_ENABLE_INQ_AND_PAGE, meaning that Inquiry Scan is enabled and Page Scan is enabled.

4.10 CSR_BT_CM_WRITE_PAGESCAN_SETTINGS

Primitives \ Parameters	type	appHandle	interval	window	resultCode	resultSupplier
CSR_BT_CM_WRITE_PAGESCAN_SETTINGS_REQ	✓	✓	✓	✓		
CSR_BT_CM_WRITE_PAGESCAN_SETTINGS_CFM	✓				✓	✓

Table 10: CSR_BT_CM_WRITE_PAGESCAN_SETTINGS Primitives

Description

The Write Page Scan settings setup the Page Scan Interval and Page Scan Window timing using in page scan mode. For more information on these parameters please consult [BT].

Note that CSR Synergy Bluetooth per default is set to use the defaults recommended by [BT], and the parameters should not be changed unnecessarily.

Parameters

type	Signal identity, CSR_BT_CM_WRITE_PAGESCAN_SETTINGS_REQ/CFM.
appHandle	The identity of the calling process.
interval	The Page Scan Interval defines the time between consecutive page scans. The value is given in slots, where 1 slot = 0.625 msec. Valid range is 0x0012 to 0x1000, and only even numbers are allowed.
window	The Page Scan Window defines how long a page scan lasts. The value is given in slots, where 1 slot = 0.625 msec. Valid range is 0x0011 to 0x1000, but the value must be less than or equal to the page scan interval.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.11 CSR_BT_CM_WRITE_PAGESCAN_TYPE

Primitives \ Parameters	type	appHandle	scanType	resultCode	resultSupplier
CSR_BT_CM_WRITE_PAGESCAN_TYPE_REQ	✓	✓	✓		
CSR_BT_CM_WRITE_PAGESCAN_TYPE_CFM	✓			✓	✓

Table 11: CSR_BT_CM_WRITE_PAGESCAN_TYPE Primitives

Description

The Write Page Scan Type setup the Page Scan Type to either “normal mode” (slower, lower power usage) or “interlaced mode” (faster, uses more power). For more information on this parameters please consult [BT].

Note that CSR Synergy Bluetooth per default is set to use the “normal mode”, as recommended by [BT], and the parameter should not be changed unnecessarily.

Parameters

type	Signal identity, CSR_BT_CM_WRITE_PAGESCAN_SETTINGS_REQ/CFM.
appHandle	The identity of the calling process.
scanType	The Page Scan Type defines the mode to use, where the allowed values are: **HCI_SCAN_TYPE_LEGACY** (0x00) is normal scan type. **HCI_SCAN_TYPE_INTERLAVED** (0x01) is interlaced scan type.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.12 CSR_BT_CM_WRITE_INQUIRYSCAN_SETTINGS

Primitives \ Parameters	type	appHandle	interval	window	resultCode	resultSupplier
CSR_BT_CM_WRITE_INQUIRYSCAN_SETTINGS_REQ	✓	✓	✓	✓		
CSR_BT_CM_WRITE_INQUIRYSCAN_SETTINGS_CFM	✓				✓	✓

Table 12: CSR_BT_CM_WRITE_INQUIRYSCAN_SETTINGS Primitives

Description

The Write Inquiry Scan settings setup the Inquiry Scan Interval and Inquiry Scan Window timing. For more information on these parameters please consult [BT].

Note that CSR Synergy Bluetooth per default is set to use the defaults recommended by [BT], and the parameters should not be changed unnecessarily.

Parameters

type	Signal identity, CSR_BT_CM_WRITE_INQUIRYSCAN_SETTINGS_REQ/CFM.
appHandle	The identity of the calling process.
interval	The Inquiry Scan Interval defines the time between consecutive inquiry scans. The value is given in slots, where 1 slot = 0.625 msec. Valid range is 0x0012 to 0x1000, and only even numbers are allowed.
window	The Inquiry Scan Window defines how long an inquiry scan lasts. The value is given in slots, where 1 slot = 0.625 msec. Valid range is 0x0011 to 0x1000, but the value must be less than or equal to the page scan interval.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.13 CSR_BT_CM_WRITE_INQUIRYSCAN_TYPE

Primitives \ Parameters	type	appHandle	scanType	resultCode	resultSupplier
CSR_BT_CM_WRITE_INQUIRYSCAN_TYPE_REQ	✓	✓	✓		
CSR_BT_CM_WRITE_INQUIRYSCAN_TYPE_CFM	✓			✓	✓

Table 13: CSR_BT_CM_WRITE_INQUIRYSCAN_TYPE Primitives

Description

The Write Inquiry Scan Type setup the Inquiry Scan Type to either “normal mode” (slower, lower power usage) or “interlaced mode” (faster, uses more power). For more information on this parameters please consult [BT].

Note that CSR Synergy Bluetooth per default is set to use the “normal mode”, as recommended by [BT], and the parameter should not be changed unnecessarily.

Parameters

type	Signal identity, CSR_BT_CM_WRITE_INQUIRYSCAN_SETTINGS_REQ/CFM.
appHandle	The identity of the calling process.
scanType	The Inquiry Scan Type defines the mode to use, where the allowed values are: **HCI_SCAN_TYPE_LEGACY** (0x00) is normal scan type. **HCI_SCAN_TYPE_INTERLAVED** (0x01) is interlaced scan type.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.14 CSR_BT_CM_CONNECTABLE

Primitives \ Parameters	type	appHandle	connectAble
CSR_BT_CM_CONNECTABLE_REQ	✓	✓	✓

Table 14: CSR_BT_CM_CONNECTABLE Primitives

Description

This command writes the value for the connectable parameter, which controls whether or not the calling process will be informed about rejected incoming RFCOMM connection attempts, see section 4.15.

Parameters

type	Signal identity, CSR_BT_CM_CONNECTABLE_REQ.
appHandle	The identity of the calling process.
connectable	If set to TRUE the calling process will be informed about rejected RFCOMM connections.

4.15 CSR_BT_CM_REJECT_RFC_CONNECTION

Primitives \ Parameters	type	deviceAddr
CSR_BT_CM_REJECT_RFC_CONNECTION_IND	✓	✓

Table 15: CSR_BT_CM_REJECT_RFC_CONNECTION Primitives

Description

Report the Bluetooth® address of the peer device which connection attempt has been rejected. Please notice that this application will only be informed if a CSR_BT_CM_CONNECTABLE_REQ is sent previously, see section 4.14.

Parameters

type	Signal identity, CSR_BT_CM_REJECT_RFC_CONNECTION_IND.
deviceAddr	The Bluetooth® address of the peer device which connection attempt has been rejected.

4.16 CSR_BT_CM_ENABLE_DUT_MODE

Primitives \ Parameters	type	appHandle	resultCode	resultSupplier	stepNumber
CSR_BT_CM_ENABLE_DUT_MODE_REQ	✓	✓			
CSR_BT_CM_ENABLE_DUT_MODE_CFM	✓		✓	✓	✓

Table 16: CSR_BT_CM_ENABLE_DUT_MODE Primitives

Description

This command will enable the Device under Test mode on the BlueCore chip. The confirm message has two parameters - the status of the operation and the step number indicating how many of the steps that are being handled. The Enable device under test consists of 3 steps in case of success. To disable and exit the Device under Test mode a reset signal must be sent.

Parameters

type	Signal identity, CSR_BT_CM_ENABLE_DUT_MODE_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
stepNumber	Is indicating how many steps the request has passed. When passing all steps the number is 3.

4.17 CSR_BT_CM_DISABLE_DUT_MODE

Primitives \ Parameters	type	appHandle	resultCode	resultSupplier
CSR_BT_CM_DISABLE_DUT_MODE_REQ	✓	✓		
CSR_BT_CM_DISABLE_DUT_MODE_CFM	✓		✓	✓

Table 17: CSR_BT_CM_DISABLE_DUT_MODE Primitives

Description

This command will disable the Device under Test mode in the CM module. This request will only disable the DUT mode in the CM; not on the chip. To disable and exit the Device under Test mode the application shall send a reset signal. This request ensures that the CM, application and chip are synchronized with regards to the Device under Test status.

Parameters

type	Signal identity, CSR_BT_CM_DISABLE_DUT_MODE_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.18 CSR_BT_CM_SDC_SEARCH

Parameters / Primitives	type	appHandle	deviceAddr	*serviceList	serviceListSize	service	*serviceHandleList	serviceHandleListCount	localServerChannel
CSR_BT_CM_SDC_SEARCH_REQ	✓	✓	✓	✓	✓				
CSR_BT_CM_SDC_SEARCH_IND	✓		✓			✓	✓	✓	✓
CSR_BT_CM_SDC_SEARCH_CFM	✓		✓						✓

Table 18: CSR_BT_CM_SDC_SEARCH Primitives

Description

To start a session for discovering services on a peer device, the application sends a CSR_BT_CM_SDC_SEARCH_REQ, with a list of services to search for.

When the CM has finished searching for all the services included in the serviceList it either returns a CSR_BT_CM_SDC_SEARCH_CFM or a CSR_BT_CM_SDC_CLOSE_IND signal.

- The CM returns the CSR_BT_CM_SDC_SEARCH_CFM if it finds at least one of the services defined in the serviceList, e.g. the initiator has already received at least one CSR_BT_CM_SDC_SEARCH_IND signal
- The CM returns a CSR_BT_CM_SDC_CLOSE_IND signal if none of the services are defined in the serviceList

If the CM returns a CSR_BT_CM_SDC_SEARCH_CFM signal, the initiator either requests to end the Service Search procedure by sending a CSR_BT_CM_SDC_CLOSE_REQ, see section 4.20, or requests to find an attribute value, see section 4.17.

Parameters

type	Signal identity, CSR_BT_CM_SDC_SEARCH_REQ/IND/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
deviceAddr	The Bluetooth® address of the peer device to be searched.
*serviceList	A list of services to search for. A service is defined as Universal Unique Identifiers (UUID), and their mnemonic and numeric values are defined in [BT12].
serviceListSize	The number of UUID in the serviceList.
service	The numeric value of the found service (UUID).
*serviceHandleList	Pointer to a list of service handles. A service handle, representing the identity of a service record on the peer device.
serviceHandleListCount	The number of service handles, and hereby the number of service records which match the service found.
localServerChannel	For internal use only, must be ignored.

One of the functions:

*void CsrBtCmSdcSearchReqSend (CsrSchedQid appHandle, CsrBtDeviceAddr deviceAddr, CsrBtUuid32 *serviceList,*
CsrUint8 serviceListSize);

void CsrBtCmSdcSearchExtReqSend (CsrSchedQid appHandle, CsrBtDeviceAddr deviceAddr,
*CsrBtUuid32 *serviceList, CsrUint8 serviceListSize);*

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_SDC_SEARCH_REQ primitive to the Connection Manager.

The difference of the above two functions are the definition of when the given list of UUID, defined in '*serviceList', must be considered valid.

- *CsrBtCmSdcSearchReqSend:* The UUID's are only consider valid, if and only if the specified UUID is contained under the Service Class ID List attribute
- *CsrBtCmSdcSearchExtReqSend:* The UUID's are consider valid, if the UUID is contained within any of the service record's attribute values.

4.19 CSR_BT_CM_SDC_ATTRIBUTE

Primitives \ Parameters	type	serviceHandle	attributeIdentifier	upperRangeAttributeIdentifier	maxBytesToReturn	*attributeList	attributeListSize	resultCode	resultSupplier	localServerChannel	deviceAddr
CSR_BT_CM_SDC_ATTRIBUTE_REQ	✓	✓	✓	✓	✓						
CSR_BT_CM_SDC_ATTRIBUTE_CFM	✓					✓	✓	✓	✓	✓	✓

Table 19: CSR_BT_CM_SDC_ATTRIBUTE Primitives

Description

After receiving a CSR_BT_CM_SDC_SEARCH_CFM signal, the initiator can start a search for an attribute value for services available on the remote device, by sending a CSR_BT_CM_SDC_ATTRIBUTE_REQ. After receiving a CSR_BT_CM_SDC_ATTRIBUTE_CFM signal the application can either request for another attribute value (this can be repeated until the initiator has obtained all the attribute values it is interested in), or it can end the Service Search procedure by sending a CSR_BT_CM_SDC_CLOSE_REQ signal.

Parameters

type	Signal identity, CSR_BT_CM_SDC_ATTRIBUTE_REQ/CFM.
serviceHandle	The service record handle of the service record being queried.
attributeIdentifiere	The attribute identifier to search for. The attribute identifiers codes numeric IDs are defined in [BT12]. Note if the function *CsrBtCmSdcAttributeRangeReqSend* is used this parameter specifies the beginning attribute identifier of a range.
upperRangeAttributeIdentifier	Specifies the ending attribute identifier of a range. Note this parameter is only valid if the function *CsrBtCmSdcAttributeRangeReqSend* is used otherwise it is set to 0….
maxBytesToReturn	The maximum number of attribute bytes to be returned. Range 0x0007 to 0x0046.
*attributeList	Data element sequence of the attribute ID. Please note that the initiator must **always** CsrPfree() this pointer in order to prevent a memory leak.
attributeListSize	The size of the attribute list in bytes.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
localServerChannel	For internal use only, must be ignored.

deviceAddr	The Bluetooth® address of the peer device to be searched for an attribute. Please note that this must be the same as the one used in the CSR_BT_CM_SDC_SEARCH_REG signal.

One of the functions:

void CsrBtCmSdcAttributeReqSend (CsrBtUuid32 serviceHandle, CsrUint16 attributeIdentifier, CsrUint16 maxBytesToReturn);

void CsrBtCmSdcAttributeRangeReqSend (CsrBtUuid32 serviceHandle, CsrUint16 attributeIdentifier, CsrUint16 upperRangeAttributeIdentifier, CsrUint16 maxBytesToReturn);

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_SDC_ATTRIBUTE_REQ primitive to the Connection Manager.

The difference of the above two functions is.

- *CsrBtCmSdcAttributeReqSend:* Only allows one attribute to be read at the time
- *CsrBtCmSdcAttributeRangeReqSend:* Allows a range of attributes to be read. Note that all attributes can be read by specifying a range of 0x0000-0xFFFF.

4.20 CSR_BT_CM_SDC_CLOSE

Primitives \ Parameters	type	appHandle	resultCode	resultSupplier	localServerChannel	deviceAddr
CSR_BT_CM_SDC_CLOSE_REQ	✓	✓				
CSR_BT_CM_SDC_CLOSE_IND	✓		✓	✓	✓	✓

Table 20: CSR_BT_CM_SDC_CLOSE Primitives

Description

The CSR_BT_CM_SDC_CLOSE_REQ signal requests to close the SDC channel, which is opened with the CSR_BT_CM_SDC_SEARCH_REQ command. When receiving a CSR_BT_CM_SDC_CLOSE_IND signal, then the SDC channel is closed and the Service Search procedure is finished.

Parameters

type	Signal identity, CSR_BT_CM_SDC_CLOSE_REQ/IND.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
localServerChannel	For internal use only, must be ignored.
deviceAddr	The Bluetooth® address of the peer device to be closed. Please note that this must be the same as the one used in the CSR_BT_CM_SDC_SEARCH_REG signal.

4.21 CSR_BT_CM_READ_REMOTE_EXT_FEATURES

Primitives \ Parameters	type	appHandle	pageNum	bd_addr	resultCode	resultSupplier	maxPageNum	extLmpFeatures
CSR_BT_CM_READ_REMOTE_EXT_FEATURES_REQ	✓	✓	✓	✓				
CSR_BT_CM_READ_REMOTE_EXT_FEATURES_CFM	✓		✓	✓	✓	✓	✓	✓

Table 21: CSR_BT_CM_READ_REMOTE_EXT_FEATURES Primitives

Description

With the CSR_BT_CM_READ_REMOTE_EXT_FEATURES_REQ signal it is possible to read the extended features of a remote device, e.g. during the connect procedure. The CSR_BT_CM_READ_REMOTE_EXT_FEATURES_CFM signal returns the result of the request for remote extended features. In the event that the remote device is older than Bluetooth version 1.2, the pageNum and maxPageNum values are both set to 0.

Parameters

type	Signal identity, CSR_BT_CM_READ_REMOTE_EXT_FEATURES_REQ /CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
pageNum	Is the page number of the extended features being requested. Page 0 returns the normal features, while pages above 0 contain the extended features.
bd_addr	The device address of the device, which extended features are requested.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
maxPageNum	Returns the maximum number of pages the remote device supports.
extLmpFeatures	Returns a bit mask of the features the remote device supports. See [BT].

4.22 CSR_BT_CM_SET_AFH_CHANNEL_CLASS

Primitives \ Parameters	type	appHandle	map[10]	resultCode	resultSupplier
CSR_BT_CM_SET_AFH_CHANNEL_CLASS_REQ	✓	✓	✓		
CSR_BT_CM_SET_AFH_CHANNEL_CLASS_CFM	✓			✓	✓

Table 22: CSR_BT_CM_SET_AFH_CHANNEL_CLASS Primitives

Description

The CSR_BT_CM_SET_AFH_CHANNEL_CLASS_REQ signal is used for specifying a channel classification e.g. based on local information held by the host. The channel classification written with this command will remain valid until it has been overwritten with another CSR_BT_CM_SET_AFH_CHANNEL_CLASS_REQ or the Bluetooth device has been reset.

Please note that this signal can be issued from different applications. In this case the CM will OR the map[10] parameters together, e.g. if one application set channel x to bad and another set channel y to bad, then will both channel x and y to set to bad. Similar, if two different applications previously has set channel x to bad and one of the application sets channel x to unknown again, then it will still be set to bad, because there is still one application that has set channel x to bad.

Parameters

type	Signal identity, CSR_BT_CM_SET_AFH_CHANNEL_CLASS_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
map[10]	This parameter is an 80 bit field, where the most significant bit is reserved and shall be set to 0. The *nth* field (in the range 0 to 78) contains the value for channel *n*: Channel *n* is bad = 0 Channel *n* is unknown = 1 See [BT].
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

The function:

*void CsrBtCmSetAfhChannelClassReqSend (CsrSchedQid thePhandle, CsrUint8 * theMap)*

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_SET_AFH_CHANNEL_CLASS_REQ primitive to the Connection Manager.

4.23 CSR_BT_CM_READ_AFH_CHANNEL_ASSESSMENT_MODE

Primitives \ Parameters	type	appHandle	classMode	resultCode	resultSupplier
CSR_BT_CM_READ_AFH_CHANNEL_ASSESSMENT_MODE_REQ	✓	✓			
CSR_BT_CM_READ_AFH_CHANNEL_ASSESSMENT_MODE_CFM	✓		✓	✓	✓

Table 23: CSR_BT_CM_READ_AFH_CHANNEL_ASSESSMENT_MODE Primitives

Description

The CSR_BT_CM_READ_AFH_CHANNEL_ASSESSMENT_MODE_REQ signal reads the status of the controller's channel assessment scheme. The result is returned in CSR_BT_CM_READ_AFH_CHANNEL_ASSESSMENT_MODE_CFM and indicates if the channel assessment scheme is enabled or disabled.

Parameters

type	Signal identity, CSR_BT_CM_READ_AFH_CHANNEL_ASSESSMENT_MODE_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
classMode	Returns FALSE if the controller's channel assessment mode is disabled and TRUE if it is enabled.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

The function:

void CsrBtCmReadAfhChannelAssessmentModeReqSend (CsrSchedQid thePhandle)

defined in csr_bt_cm.lib.h, build and send the CSR_BT_CM_READ_AFH_CHANNEL_ASSESSMENT_MODE_REQ primitive to the Connection Manager.

4.24 CSR_BT_CM_WRITE_AFH_CHANNEL_ASSESSMENT_MODE

Primitives \ Parameters	type	appHandle	classMode	resultCode	resultSupplier
CSR_BT_CM_WRITE_AFH_CHANNEL_ASSESSMENT_MODE_REQ	✓	✓	✓		
CSR_BT_CM_WRITE_AFH_CHANNEL_ASSESSMENT_MODE_CFM	✓			✓	✓

Table 24: CSR_BT_CM_WRITE_AFH_CHANNEL_ASSESSMENT_MODE Primitives

Description

The CSR_BT_CM_WRITE_AFH_CHANNEL_ASSESSMENT_MODE_REQ signal is used for activating or deactivate the channel assessment mode.

Parameters

type	Signal identity, CSR_BT_CM_WRITE_AFH_CHANNEL_ASSESSMENT_MODE _REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
classMode	If the classMode parameter is set to FALSE this signal will disable the channel assessment mode. If classMode is set to TRUE the channel assessment mode will be enabled.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

The function:

void CsrBtCmWriteAfhChannelAssessmentModeReqSend (CsrSchedQid thePhandle, CsrUint8 theClassMode)

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_WRITE_AFH_CHANNEL_ASSESSMENT_MODE_REQ primitive to the Connection Manager.

4.25 CSR_BT_CM_READ_LOCAL_EXT_FEATURES

Primitives \ Parameters	type	appHandle	pageNum	resultCode	resultSupplier	maxPageNum	extLmpFeatures
CSR_BT_CM_READ_LOCAL_EXT_FEATURES_REQ	✓	✓	✓				
CSR_BT_CM_READ_LOCAL_EXT_FEATURES_CFM	✓		✓	✓	✓	✓	✓

Table 25: CSR_BT_CM_READ_LOCAL_EXT_FEATURES Primitives

Description

With the CSR_BT_CM_READ_LOCAL_EXT_FEATURES_REQ signal it is possible to read the extended features of a local Bluetooth device. The CSR_BT_CM_READ_LOCAL_EXT_FEATURES_CFM signal returns the result of the request.

Parameters

type	Signal identity, CSR_BT_CM_READ_LOCAL_EXT_FEATURES_REQ /CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
pageNum	The page number of the extended features that are requested. Page 0 returns the normal features, while page numbers above 0 contains the extended features.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
maxPageNum	Returns the maximum number of pages the device supports.
extLmpFeatures	Returns a bit mask of the features supported by the device. See [BT].

The function:

void CsrBtCmReadLocalExtFeaturesReqSend (CsrSchedQid thePhandle, CsrUint8 thePageNum)

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_READ_LOCAL_EXT_FEATURES_REQ primitive to the Connection Manager.

4.26 CSR_BT_CM_READ_AFH_CHANNEL_MAP

Primitives \ Parameters	type	appHandle	bd_addr	resultCode	resultSupplier	mode	afhMap[10]
CSR_BT_CM_READ_AFH_CHANNEL_MAP_REQ	✓	✓	✓				
CSR_BT_CM_READ_AFH_CHANNEL_MAP_CFM	✓		✓	✓	✓	✓	✓

Table 26: CSR_BT_CM_READ_AFH_CHANNEL_MAP Primitives

Description

This signal will return the channel classification map for the connection specified by the bd_addr, which must be a device the host is already connected to.

Parameters

type	Signal identity, CSR_BT_CM_READ_AFH_CHANNEL_MAP_REQ /CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
bd_addr	A device address that specifies the connection for which to read the classification map.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
mode	Specifies if AFH mode is enabled or disabled. Returns TRUE if AFH is enabled and FALSE if AFH is disabled.
afhMap[10]	If AFH is enabled then this parameter contains 79 1-bit fields otherwise the contents are invalid. The *nth* such field (in the range 0 to 78) contains the value for channel *n*: Channel *n* is unused = 0 Channel *n* is used = 1

The function:

void CsrBtCmReadAfhChannelMapReqSend (CsrSchedQid thePhandle, deviceAddr_t theDeviceAddr)

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_READ_AFH_CHANNEL_MAP_REQ primitive to the Connection Manager.

4.27 CSR_BT_CM_READ_CLOCK

Primitives \ Parameters	type	appHandle	whichClock	bd_addr	resultCode	resultSupplier	clock	accuracy
CSR_BT_CM_READ_CLOCK_REQ	✓	✓	✓	✓				
CSR_BT_CM_READ_CLOCK_CFM	✓			✓	✓	✓	✓	✓

Table 27: CSR_BT_CM_READ_CLOCK Primitives

Description

The CSR_BT_CM_READ_CLOCK signal will read the estimate of the Bluetooth clock. The device has to be connected to another device before the value of the piconet clock can be read.

Parameters

type	Signal identity, CSR_BT_CM_READ_CLOCK_REQ /CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
whichClock	The whichClock parameter is used for specifying which clock that should be read. If 0 is specified it is an estimate of the local clock that will be returned. If 1 is specified the estimate of the piconet clock will be returned.
bd_addr	The device address of the other device in the piconet.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
clock	The parameter returns the local clock or the piconet clock, depending on the request.
accuracy	Returns the accuracy of the clock result.

The function:

void CsrBtCmReadClockReqSend (CsrSchedQid thePhandle,CsrUint8 theClock, deviceAddr_t theDeviceAddr)

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_READ_CLOCK_REQ primitive to the Connection Manager.

4.28 CSR_BT_CM_READ_TX_POWER_LEVEL

Parameters / Primitives	type	appHandle	deviceAddr	levelType	resultCode	resultSupplier	powerLevel	addressType
CSR_BT_CM_READ_TX_POWER_LEVEL_REQ	✓	✓	✓	✓				✓
CSR_BT_CM_READ_TX_POWER_LEVEL_CFM	✓		✓		✓	✓	✓	

Table 28: CSR_BT_CM_READ_TX_POWER_LEVEL Primitives

Description

Read the transmit power level.

Parameters

Type	Signal identity, CSR_BT_CM_READ_TX_POWER_LEVEL_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
deviceAddr	The Bluetooth® device address of the remote Bluetooth® device found during the device discovery process.
levelType	Read the current or the maximum power level. (0 = current, 1 = maximum)
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
powerLevel	The current or maximum transmit power level
addressType	Address type of 'deviceAddr' (see CSR_BT_ADDR_ defines in csr_bt_addr.h)

The functions:

void CsrBtCmReadTxPowerLevelReqSend (CsrSchedQid thePhandle, deviceAddr_t theDeviceAddr, CsrUint8 theLevelType)

void CsrBtCmReadTxPowerLevelReqSend Ex(CsrSchedQid thePhandle, deviceAddr_t theDeviceAddr, CsrUint8 theLevelType, CsrBtAddressType addressType)

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_READ_TX_POWER_LEVEL_REQ primitive to the Connection Manager.

4.29 CSR_BT_CM_GET_LINK_QUALITY

Primitives \ Parameters	type	appHandle	deviceAddr	resultCode	resultSupplier	linkQuality
CSR_BT_CM_GET_LINK_QUALITY_REQ	✓	✓	✓			
CSR_BT_CM_GET_LINK_QUALITY_CFM	✓		✓	✓	✓	✓

Table 29: CSR_BT_CM_GET_LINK_QUALITY Primitives

Description

Request to get the link quality parameter.

Parameters

type — Signal identity, CSR_BT_CM_GET_LINK_QUALITY_REQ/CFM.

appHandle — The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.

deviceAddr — The Bluetooth® device address of the remote Bluetooth® device found during the device discovery process.

resultCode — The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.

resultSupplier — This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

linkQuality — The current quality of the link. Range 0x00 – 0xFF. The Higher value, the better the link quality is.

The function:

void CsrBtCmGetLinkQualityReqSend (CsrSchedQid thePhandle, deviceAddr_t theDeviceAddr)

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_GET_LINK_QUALITY_REQ primitive to the Connection Manager.

4.30 CSR_BT_CM_READ_RSSI

Parameters / Primitives	type	appHandle	deviceAddr	resultCode	resultSupplier	rssi	addressType
CSR_BT_CM_READ_RSSI_REQ	✓	✓	✓				✓
CSR_BT_CM_READ_RSSI_CFM	✓		✓	✓	✓	✓	

Table 30: CSR_BT_CM_READ_RSSI Primitives

Description

Request to read the link RSSI value.

Parameters

Type — Signal identity, CSR_BT_CM_READ_RSSI_REQ/CFM.

appHandle — The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.

deviceAddr — The Bluetooth® device address of the remote Bluetooth® device found during the device discovery process.

resultCode — The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.

resultSupplier — This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

rssi — The link RSSI value

addressType — Address type of 'deviceAddr' (see CSR_BT_ADDR_ defines in csr_bt_addr.h)

The functions:

void CsrBtCmReadRssiReqSend (CsrSchedQid thePhandle, deviceAddr_t theDeviceAddr)

void CsrBtCmReadRssiReqSendEx (CsrSchedQid thePhandle, deviceAddr_t theDeviceAddr, CsrBtAddressType addrType)

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_READ_RSSI_REQ primitive to the Connection Manager.

4.31 CSR_BT_CM_READ_LOCAL_NAME

Primitives \ Parameters	type	phandle	localName
CSR_BT_CM_READ_LOCAL_NAME_REQ	✓	✓	
CSR_BT_CM_READ_LOCAL_NAME_CFM	✓		✓

Table 31: CSR_BT_CM_READ_LOCAL_NAME Primitives

Description

Read the local name of the device.

Parameters

type — Signal identity, CSR_BT_CM_READ_LOCAL_NAME_REQ /CFM.

phandle — The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.

localName — Utf-8 string pointer containing the name of the local device.

The function:

void CsrBtCmReadLocalNameReqSend (CsrSchedQid thePhandle)

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_READ_LOCAL_NAME_REQ primitive to the Connection Manager.

4.32 CSR_BT_CM_WRITE_PAGE_TO

Primitives \ Parameters	type	appHandle	pageTimeout	resultCode	resultSupplier
CSR_BT_CM_WRITE_PAGE_TO_REQ	✓	✓	✓		
CSR_BT_CM_WRITE_PAGE_TO_CFM	✓			✓	✓

Table 32: CSR_BT_CM_WRITE_PAGE_TO Primitives

Description

Writes the value of the page timeout window

Parameters

type	Signal identity, CSR_BT_CM_WRITE_PAGE_TO_REQ /CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
pageTimeout	Page timeout measured in number of baseband slots. Interval length = N * 0,625msec. Range 0x0001 – 0xFFFF.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

The function:

void CsrBtCmWritePageToReqSend (CsrSchedQid thePhandle, CsrUint16 thePageTimeout)

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_WRITE_PAGE_TO_REQ primitive to the Connection Manager.

4.33 CSR_BT_CM_SDC_UUID128_SEARCH

Primitives \ Parameters	type	appHandle	deviceAddr	*serviceList	serviceListSize	service	*serviceHandleList	serviceHandleListCount	localServerChannel
CSR_BT_CM_SDC_UUID128_SEARCH_REQ	✓	✓	✓	✓	✓				
CSR_BT_CM_SDC_UUID128_SEARCH_IND	✓		✓			✓	✓	✓	✓

Table 33: CSR_BT_CM_SDC_UUID128_SEARCH Primitives

Description

To start a session for discovering of 128bit services (like syncML) on a peer device, the application sends a CSR_BT_CM_SDC_UUID128_SEARCH_REQ, with a list of services to search for.

When the CM has finished searching for all the services included in the serviceList it either returns a CSR_BT_CM_SDC_SEARCH_CFM, see section 4.17, or a CSR_BT_CM_SDC_CLOSE_IND signal, see section 4.20.

- The CM returns the CSR_BT_CM_SDC_SEARCH_CFM if it finds at least one of the services defined in the serviceList, e.g. the initiator has already received at least one CSR_BT_CM_SDC_UUID128_SEARCH_IND signal
- The CM returns a CSR_BT_CM_SDC_CLOSE_IND signal if none of the services are defined in the serviceList

If the CM returns a CSR_BT_CM_SDC_SEARCH_CFM signal, the initiator can either request to end the UUID128 Service Search procedure by sending a CSR_BT_CM_SDC_CLOSE_REQ, see section 4.20, or request to find an attribute value, see section 4.17.

Parameters

type	Signal identity, CSR_BT_CM_SDC_UUID128_SEARCH_REQ/IND.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
deviceAddr	The Bluetooth® address of the peer device to be searched.
*serviceList	A list of 128bit services to search for. A service is defined as Universal Unique Identifiers (UUID), and their mnemonic and numeric values are defined in [BT12].
serviceListSize	The number of 128bit UUID in the serviceList.
service	The numeric value of the found 128 bit service (UUID).
*serviceHandleList	Pointer to a list of service handles. A service handle, representing the identity of a service record on the peer device.
serviceHandleListCount	The number of service handles, and hereby the number of service records which match the service found.
localServerChannel	For internal use only, must be ignored.

Example

A 128bit Service search can be requested by using the function *CsrBtCmSdcUuid128SearchReqSend* which is defined in csr_bt_cm_lib.h.

```
uuid128_t syncML  = {0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x10, 0x00, 0x80, 0x00, 0x00, 0x02, 0xEE, 0x00, 0x00, 0x02 };

CsrBtCmSdcUuid128SearchReqSend(APP_QUEUE, deviceAddr, syncML, 1);
```

In this example a search only for syncML is requested, and the Bluetooth device address has been previously obtained, probably by the use of Inquiry.

4.34 CSR_BT_CM_SDC_CANCEL_SEARCH

Primitives \ Parameters	type	apphandle	deviceAddr	typeToCancel
CSR_BT_CM_SDC_CANCEL_SEARCH_ REQ	✓	✓	✓	✓

Table 34: CSR_BT_CM_SDC_CANCEL_SEARCH Primitives

Description

The application can cancel a CSR_BT_CM_SDC_SEARCH_REQ or a CSR_BT_CM_SDC_UUID128_SEARCH_REQ by sending a CSR_BT_CM_SDC_CANCEL_SEARCH_REQ. If the CSR_BT_CM_SDC_SEARCH_REQ or the CSR_BT_CM_SDC_UUID128_SEARCH_REQ is cancelled the application will receive a CSR_BT_CM_SDC_CLOSE_IND.

Please notice that CSR_BT_CM_SDC_CANCEL_SEARCH_REQ is used for cancelling both a CSR_BT_CM_SDC_SEARCH_REQ and a CSR_BT_CM_SDC_UUID128_SEARCH_REQ. Therefore the application **must** use the library function:

void CsrBtCmSdcCancelSearchReqSend (CsrSchedQid appHandle, deviceAddr_t deviceAddr)

to cancel a CSR_BT_CM_SDC_SEARCH_REQ.

and use the library function:

void CsrBtCmSdcCancelUuid128SearchReqSend (CsrSchedQid appHandle, deviceAddr_t deviceAddr)

to cancel a CSR_BT_CM_SDC_UUID128_SEARCH_REQ.

These functions are defined in csr_bt_cm_lib.h.

Parameters

type	Signal identity, CSR_BT_CM_SDC_CANCEL_SEARCH_REQ.
apphandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.
deviceAddr	The Bluetooth® address previously used in the CSR_BT_CM_SDC_SEARCH_REG or in the CSR_BT_CM_SDC_UUID128_SEARCH_REQ depending on which search procedure to cancel
typeToCancel	Is a private variable used by CM libraries functions, *CsrBtCmSdcCancelSearchReqSend* and *CsrBtCmSdcCancelUuid128SearchReqSend*, which is defined in csr_bt_cm_lib.h.

4.35 CSR_BT_CM_ROLE_DISCOVERY

Primitives \ Parameters	type	pHandle	deviceAddr	role
CSR_BT_CM_ROLE_DISCOVERY_REQ	✓	✓	✓	
CSR_BT_CM_ROLE_DISCOVERY_CFM	✓		✓	✓

Table 35: CSR_BT_CM_ROLE_DISCOVERY Primitives

Description

Request the current role (master or slave) of the local device in an ACL connection.

The function:

void CsrBtCmRoleDiscoveryReqSend (CsrSchedQid appHandle, deviceAddr_t deviceAddr)

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_ROLE_DISCOVERY_REQ primitive to the Connection Manager.

Parameters

type — Signal identity, CSR_BT_CM_ROLE_DISCOVERY _REQ/CFM.

appHandle — The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to pHandle.

deviceAddr — The Bluetooth® address of a connected peer device, for which to check master/slave roles.

role — Role of the local device in the specified connection. Values are defined in csr_bt_profiles.h, and may be:

MASTER_ROLE

SLAVE_ROLE

UNDEFINED_ROLE: The role discovery failed for some reason.

4.36 CSR_BT_CM_WRITE_LINK_POLICY/CSR_BT_CM_WRITE_LINK_POLICY_ERROR

Primitives \ Parameters	type	appHandle	deviceAddr	setupLinkPolicySetting	linkPolicySetting	*sniffSettings	*parkSettings	resultCode	resultSupplier
CSR_BT_CM_WRITE_LINK_POLICY_REQ	✓	✓	✓	✓	✓	✓	✓		
CSR_BT_CM_WRITE_LINK_POLICY_ERROR_IND	✓		✓					✓	✓

Table 36: CSR_BT_CM_WRITE_LINK_POLICY/ CSR_BT_CM_WRITE_LINK_POLICY_ERROR Primitives

Description

This command can be used for controlling which policy settings - i.e. sniff/park are allowed. The default policy settings will be taken from the defines:

CSR_BT_SNIFF_MAX_TIME_INTERVAL
CSR_BT_SNIFF_MIN_TIME_INTERVAL
CSR_BT_SNIFF_ATTEMPT
CSR_BT_SNIFF_TIMEOUT
CSR_BT_PARK_MAX_TIME_INTERVAL
CSR_BT_PARK_MIN_TIME_INTERVAL

which can be found in csr_bt_usr_config.h.

Parameters

type	Signal identity, CSR_BT_CM_WRITE_LINK_POLICY_REQ/ CSR_BT_CM_WRITE_LINK_POLICY_ERROR_IND.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
deviceAddr	The Bluetooth® device address of the remote Bluetooth® device which the link policy must be changed to. Please note that there must be a link established to the remote Bluetooth® device otherwise the error message CSR_BT_CM_WRITE_LINK_POLICY_ERROR_IND is returned. After the link is released the Connection Manager will again use the default link policy settings. A Bluetooth® device address of 0 will change the CSR Synergy Bluetooth default link policy settings, e.g. if *linkPolicySetting* parameter is set to ENABLE_SNIFF the local device will as default only enable local support for sniff mode for all remote Bluetooth® devices. Changing the default link policy will *not* modify link policies for already established links. Only new links will use the new default link policy.
setupLinkPolicySetting	If TRUE the value of the parameter *linkPolicySetting* will be valid.
linkPolicySetting	The following values are valid and defined in csr_bt_hci.h: DISABLE_ALL_LM_MODES (0x0000). Disable local support of sniff and park mode. ENABLE_SNIFF (0x0004). Enable local support of sniff mode.

	ENABLE_PARK (0x0008). Enable local support of park mode. ENABLE_SNIFF | ENABLE_PARK (0x000C). Enable local support of sniff and park mode. ENABLE_MS_SWITCH (0x0001) Enables master/slave switches. This flag is only usable when the Bluetooth address is zero. The default setting for the linkPolicySetting parameter is: (ENABLE_SNIFF | ENABLE_PARK) which enables local support for sniff and park mode. Please note that if the link is in sniff mode and sniff mode is disabled the link will be forced out of sniff mode. Likewise, if the link is in park mode and park mode is disabled the link will be forced out of park mode.
*sniffSettings	The Sniff interval parameters. Please note that the Sniff interval parameter is not changed while the link is in sniff mode. The new sniff interval parameter will not be used until the next time the local device sets the link in sniff mode. Please note that these settings are only used when the local device sets the link in sniff mode.
*parkSettings	The park interval parameters. Please note that the park interval parameter is not change while the link is in park mode. The new park interval parameter will not be used until the next time the local device sets the link in park mode. Please note that these settings are only used when the local device sets the link in park mode.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If eg. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

The function:

*void CsrBtCmWriteLinkPolicyReqSend (deviceAddr_t theDeviceAddr, link_policy_settings_t theLinkPolicySetting, PARK_SETTINGS_T *theParkSettings, CsrBool setupLinkPolicySetting, SNIFF_SETTINGS_T *theSniffSettings)*

defined in csr_bt_cm_lib.h, build and sends the CSR_BT_CM_WRITE_LINK_POLICY_REQ primitive to the Connection Manager.

Please note that if the parameters **theParkSettings, *theSniffSettings* is set to NULL the Sniff/Park intervals are not changed.

The structure of the *sniffSettings* parameter is defined in csr_bt_dm_prim.h as:

```
typedef struct
{
  CsrUint16  max_interval;
  CsrUint16     min_interval;
  CsrUint16   attempt;
  CsrUint16  timeout;
} SNIFF_SETTINGS_T;
```

where, the *max_interval* and *min_interval* parameters specify the requested acceptable maximum and minimum periods in the Sniff Mode. The *min_interval* shall not be greater than the *max_interval* and the *max_interval* must be less than the Link Supervision Timeout configuration (see section 4.4) to ensure that the link supervision timer don't consider the sniff period as a link lost.

The sniff interval defines the amount of time between each consecutive sniff period. The range is: 0x0002 to 0xFFFE (a time range from 1.25 msec to 40.9 sec). Please note that **only** even values are valid. The mandatory range is: 0x0006 to 0x0540.

The *attempt* parameter defines the number of Baseband received slots for sniff attempts and the *timeout* parameter defines the number of Baseband received slots for sniff timeout. The range for the *attempt* parameter is: 0x0001 to 0x7FFF and the range for the *timeout* parameter is 0x0000 to 0x7FFF. For more information about the *attempt* and *timeout* parameters, please refer to [BT].

The structure of the *parkSettings* parameter is defined in csr_bt_dm_prim.h as:

```
typedef struct
{
  CsrUint16  max_interval;
  CsrUint16     min_interval;
  CsrUint16   park_idle_time;
} PARK_SETTINGS_T;
```

where, the *max_interval* and *min_interval* parameters specify the acceptable length of the interval between beacons. The *max_interval* parameter specifies the acceptable longest length of the interval between beacons. The *min_interval* parameter specifies the acceptable shortest length of the interval between beacons. Therefore, the *min_interval* parameter cannot be larger than the *max_interval parameter.*

The *park_idle_time* parameter is reserved for future use and will always be set to 0.

Example

In this example the default link policy is changed so the local device only supports Active and Sniff mode. Note that in this example the sniff intervals are changed and the park interval keeps its default settings.

```
deviceAddr_t            devAddr;

SNIFF_SETTINGS_T        sniffSettings;


bd_addr_zero(&devAddr);

sniffSettings.max_interval   = 600;

sniffSettings.min_interval   = 400;

sniffSettings.attempt        = 100;

sniffSettings.timeout        = 400;


CsrBtCmWriteLinkPolicyReqSend(APP_QUEUE, devAddr, ENABLE_SNIFF, NULL, TRUE, &sniffSettings),
```

4.37 CSR_BT_CM_READ_LINK_POLICY

Parameters / Primitives	type	appHandle	deviceAddr	resultCode	resultSupplier	actualMode	linkPolicySetting	sniffSettings	parkSettings
CSR_BT_CM_READ_LINK_POLICY_REQ	✓	✓	✓						
CSR_BT_CM_READ_LINK_POLICY_CFM	✓		✓	✓	✓	✓	✓	✓	✓

Table 37: CSR_BT_CM_READ_LINK_POLICY Primitives

Description

This command can be used for reading which policy settings - i.e. sniff/park are allowed.

The function:

void CsrBtCmReadLinkPolicyReqSend (CsrSchedQid theAppHandle, deviceAddr_t theDeviceAddr)

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_READ_LINK_POLICY_REQ primitive to the Connection Manager.

Parameters

type	The signal identity, CSR_BT_CM_READ_LINK_POLICY_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
deviceAddr	The Bluetooth® device address of the remote Bluetooth® device which the link policy must read. Please note that there must be a link established to the remote Bluetooth® device. A Bluetooth® device address of 0 will read the CSR Synergy Bluetooth default link policy settings.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
actualMode	Return the actual mode the link is in – i.e. active/sniff/park mode. Please note that if the default settings are read –i.e. using a Bluetooth® device address of 0, this parameter is invalid.
linkPolicySetting	Contains which policy settings that is currently allowed. DISABLE_ALL_LM_MODES (0x0000). Only active mode is supported by the local device.

	ENABLE_SNIFF (0x0004). Active/Sniff mode is supported by the local device. ENABLE_PARK (0x0008). Active/Park mode is supported by the local device. ENABLE_SNIFF | ENABLE_PARK (0x000C). Active/Sniff/Park is supported by the local device.
sniffSettings	The Sniff interval settings.
parkSettings	The Park interval settings.

4.38 CSR_BT_CM_EIR_UPDATE_MANUFACTURER_DATA

Primitives \ Parameters	type	appHandle	manufacturerDataSettings	manufacturerDataLength	*manufacturerData	resultCode	resultSupplier
CSR_BT_CM_EIR_UPDATE_MANUFACTURER_DATA_REQ	✓	✓	✓	✓	✓		
CSR_BT_CM_EIR_UPDATE_MANUFACTURER_DATA_CFM	✓					✓	✓

Table 38: CSR_BT_CM_EIR_UPDATE_MANUFACTURER_DATA Primitives

Description

The application can update the manufacturer specific data in the local Extended Inquiry Response (EIR) by using the CSR_BT_CM_EIR_UPDATE_MANUFACTURER_DATA_REQ primitive. A confirmation will be sent to the application requesting the update of the manufacturer data indicating whether the request was handled successfully or not.

The manufacturer specific data is global for all applications and will be removed from the local EIR if the controller or stack is reset. By default no manufacturer data will be present in the local EIR.

The function:

```
CsrBtCmEirUpdateManufacturerDataReqSend(    CsrSchedQid appHandle,
                                            CsrUint8 manufacturerDataSettings,
                                            CsrUint8  manufacturerDataLength,
                                            CsrUint8 *manufacturerData);
```

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_EIR_UPDATE_MANUFACTURER_DATA_REQ primitive to the Connection Manager. It should be noted that the function will take a copy of the data pointed to by *manufacturerData*.

Parameters

type	The signal identity, CSR_BT_CM_EIR_UPDATE_MANUFACTUER_DATA_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.

manufacturerDataSettings	Three different settings can be used for manufacturer specific EIR data: **EIR_MANUFACTURER_NOT_AVAILABLE:** This value must be used for removing the manufacturer specific data from the local EIR. By default the local EIR does not contain any manufacturer specific data, so this setting should only be used after previously updating the manufacturer data. **EIR_MANUFACTURER_PRIORITY_LOW:** All other data types (such as supported services and local name) will take precedence over manufacturer data. This is the recommended setting. **EIR_MANUFACTURER_PRIORITY_HIGH:** The manufacturer data will take precedence over all other data types in the local EIR. If this setting is used in combination with a large amount of manufacturer data, please be aware that the local name and supported services might not be available in the local EIR. This parameter must be set to one of the specified values – all other values are reserved for future use.
manufacturerDataLength	The maximum size of the manufacturer specific EIR data is specified by EIR_MANUFACTURER_DATA_MAX_SIZE (238 octets). It is recommended to keep the amount of the manufacturer specific data as low as possible to reduce power consumption of both the local and remote devices and to make sure that space is available in the local EIR for the name and the supported services. In order to remove previously set manufacturer specific data, this value must be set to 0.
*manufacturerData	A pointer to the actual data to insert into the local EIR. This value should be set to NULL if the manufacturer specific data should be removed from the local EIR.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.39 CSR_BT_CM_ WRITE_COD

Parameters / Primitives	type	appHandle	resultCode	resultSupplier	serviceClassOfDevice	majorClassOfDevice	minorClassOfDevice
CSR_BT_CM_WRITE_COD_REQ	✓	✓			✓	✓	✓
CSR_BT_CM_WRITE_COD_CFM	✓		✓	✓			

Table 39: CSR_BT_CM_WRITE_COD Primitives

Description

The application can set the class of device of the application using CSR_BT_CM_WRITE_COD_REQ. The application can choose to set either major/minor or service class of device using the following library functions respectively:

The function: *CsrBtCmWriteServiceCodReqSend (CsrSchedQid appHandleSend, CsrUint24 service);*

The function: *CsrBtCmWriteMajorMinorCcodReqSend (CsrSchedQid appHandleSend CsrUint24 major, CsrUint24 minor);*

defined in csr_bt_cm_lib.h, builds and sends the CSR_BT_CM_WRITE_COD_REQ primitive to the Connection Manager.

The default major/minor class of device bits will be taken from the define MAJOR_MINOR_DEVICE_CLASS found in csr_bt_usr_config.h if the application does not specify anything.

Parameters

type	Signal identity, CSR_BT_CM_WRITE_COD_REQ.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
serviceClassOfDevice	The service class of device value , which can be found in csr_bt_profiles.h. Only bits 13-23 of this value will be used.
majorClassOfDevice	The major class of device value , which can be found in csr_bt_profiles.h. Only bits 8-12 of this value will be used.
minorClassOfDevice	The minor class of device value , which can be found in csr_bt_profiles.h Only bits 0-7 of this value will be used.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.40 CSR_BT_CM_ READ_COD

Parameters / Primitives	type	appHandle	resultCode	resultSupplier	classOfDevice
CSR_BT_CM_READ_COD_REQ	✓	✓			
CSR_BT_CM_READ_COD_CFM	✓		✓	✓	✓

Table 40: CSR_BT_CM_WRITE_COD Primitives

Description

The application can read the Class of device of the application using CSR_BT_CM_READ_COD_REQ.

The function:

CsrBtCmReadCodReqSend (CsrSchedQid appHandleSend);

defined in csr_bt_cm_lib.h, builds and sends the CSR_BT_CM_READ_COD_REQ primitive to the Connection Manager.

Parameters

type	Signal identity, CSR_BT_CM_READ_COD_REQ.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
classOfDevice	The full class of device value (ie. bits 0-23) , The meaning of each bit can be found in profiles.h
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.41 CSR_BT_CM_READ_LOCAL_VERSION

Parameters / Primitives	type	phandle	lmpVersion
CSR_BT_CM_READ_LOCAL_VERSION_REQ	✓	✓	
CSR_BT_CM_READ_LOCAL_VERSION_CFM	✓		✓

Table 41: CSR_BT_CM_READ_LOCAL_VERSION Primitives

Description

Read the local lmp version.

The function:

CsrBtCmReadLocalVersionReqSend (CsrSchedQid thePhandle);

defined in csr_bt_cm_lib.h, builds and sends the CSR_BT_CM_READ_LOCAL_VERSION_REQ primitive to the Connection Manager.

Parameters

type	Signal identity, CSR_BT_CM_READ_LOCAL_VERSION_REQ/CFM.
phandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.
lmpVersion	The version of the local device.

4.42 CSR_BT_CM_ROLE_SWITCH_CONFIG

Primitives \ Parameters	type	config
CSR_BT_CM_ROLE_SWITCH_CONFIG_REQ	✓	✓

Table 42: CSR_BT_CM_ROLE_SWITCH_CONFIG Primitive

Description

The application can control how the Connection Manager behaves in a scatter/piconet formation by sending a CSR_BT_CM_ROLE_SWITCH_CONFIG_REQ signal.

The function:

CsrBtCmRoleSwitchConfigReqSend (CsrUint32 config);

defined in csr_bt_cm_lib.h, builds and sends the CSR_BT_CM_ ROLE_SWITCH_CONFIG_REQ primitive to the Connection Manager.

Parameters

type — Signal identity, CSR_BT_CM_ROLE_SWITCH_CONFIG_REQ.

config — Is use to set three flags which control the role switch behaviour, where one flag tells the Connection Manager (CM) if it always must try to be master or not after an ACL connection is establish. Another tells the CM if it shall try to become master or not before it initiate a Synchronous (SCO) connection, and the last flag tells the CM if it shall try to become master or not before it attempts to read a remote name.

The following values are allowed (see csr_bt_cm_prim.h), Please note as these values don't represent bitmask values the application may need to send this message multiple times in order to obtain the required setting.

- **CSR_BT_CM_ROLE_SWITCH_DEFAULT**: Default behaviour. The CM will try to do a role switch before initiating a SCO connection or if there are more than one ACL present. Note the CM will not try to become master before attempting to read a remote name, even if **CSR_BT_CM_ROLE_SWITCH_BEFORE_RNR** previously has been set.
- **CSR_BT_CM_ROLE_SWITCH_BEFORE_SCO:** The CM must try to become master before initiating a Synchronous (SCO) connection. Note this value will not change the value of the other two flags.
- **CSR_BT_CM_ROLE_SWITCH_NOT_BEFORE_SCO**: The CM shall not try to become master before initiating a Synchronous (SCO) connection. Note this value will not change the value of the other two flags.
- **CSR_BT_CM_ROLE_SWITCH_BEFORE_RNR**: The CM must try to become master before reading a peer device friendly name. Note this value will not change the value of the other two flags.
- **CSR_BT_CM_ROLE_SWITCH_NOT_BEFORE_RNR:** The CM shall not try to become master before reading a peer device friendly name. Note this value will not change the value of the other two flags.
- **CSR_BT_CM_ROLE_SWITCH_ ALWAYS_ACL:** The CM must try to become master every time an ACL connection is established. Note this value will not

change the value of the other two flags.

- **CSR_BT_CM_ROLE_SWITCH_MULTIPLE_ACL:** The CM must only try to become master if there are more than one ACL present. Note this value will not change the value of the other two flags.
- **CSR_BT_CM_ROLE_SWITCH_ALWAYS**: The CM must try to become master every time an ACL connection is established, and before it initiating a Synchronous (SCO) connection. Note the value of the flag, which controls whether or not the CM must try to become master before reading a peer device friendly name is not change.

4.43 CSR_BT_CM_SET_EVENT_MASK

Primitives \ Parameters	type	phandle	eventMask	conditionMask
CSR_BT_CM_SET_EVENT_MASK_REQ	✓	✓	✓	✓
CSR_BT_CM_SET_EVENT_MASK_CFM	✓		✓	

Table 43: CSR_BT_CM_SET_ EVENT_MASK Primitives

Description

This signal can be used for setting which extended information the application will subscribe for, and may be sent momentarily from the application.

The function:

CsrBtCmSetEventMaskReqSend (CsrSchedQid phandle, CsrUint32 eventMask, CsrUint32 conditionMask);

defined in csr_bt_cm_lib.h, builds and sends the CSR_BT_CM_SET _EVENT_MASK_REQ primitive to the Connection Manager.

Parameters

type — Signal identity, CSR_BT_CM_SET _EVENT_MASK_REQ /CFM.

phandle — The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.

eventMask — In the request message the eventMask parameter describes which extended information an application will subscribe for, and the confirm message describes which event has been set.

In the case that the application subscribes for a particular event that is already running in the Connection Manager the application will be informed of these as well, e.g. if the application subscribes for synchronous connection events and a synchronous connection is established, then the application will receive a CSR_BT_CM_SYNC_CONNECT_IND (see section 4.44) as it has just been established. Another example could be that the applications subscribe for ACL connections events while some ACL connections are already setup. In this case the application will receive a CSR_BT_CM_ACL_CONNECT_IND for each connected ACL connection (see section 4.49).

Please note that in cases where the application set the eventMask parameter to a value that it is not supported then only the supported value will be set in the confirmed message. The event mask values are defined in csr_bt_cm_prim.h.

Value	Parameter description	Reference
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_NONE	Stop subscribing for any extended information.	N/A
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_SYNCHRO	Synchronous	4.44

NOUS _CONNECTION	connection events	
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_MODE_CHANGE	Mode Change events	4.45
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_ROLE_CHANGE	Role Change events	4.46
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_LSTO_CHANGE	Link Supervision Timeout Changed event	4.47
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_BLUECORE _ INITIALIZED	BlueCore initialized event	4.48
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_ACL_CONNECTION	ACL Connection events	4.49
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_CHANNEL_TYPE	Channel type changes	4.55
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_EXT_SYNC_CONNECTION	Extended synchronous connection events	4.44
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_REMOTE_FEATURES	Read Remote Features event	4.56
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_REMOTE_VERSION	Read Remote Version event	4.7
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_A2DP_BIT_RATE	A2DP bit rate event	4.57
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_INQUIRY_PAGE_STATE	Inquiry and page state event	4.58
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_BLE_CONNECTION	BLE connection event	4.59
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_ENCRYPT_CHANGE	Encryption Change event	4.60
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_LOCAL_NAME_CHANGE	Local Name Change event	4.65
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_LOW_ENERGY	Advertising, scan and connection event	4.66, 4.67 and 4.68
CSR_BT_CM_EVENT_MASK_SUBSCRIBE_HIGH_PRIORITY_DATA	Sending high priority data.	4.69

conditionMask

In the request message the conditionMask parameter in use defines at which condition the extended information must be given to the application. The condition values are defined in in csr_bt_cm_prim.h.

Value	Parameter description
CSR_BT_CM_EVENT_MASK_COND_ALL	No condition, e.g. the extended information that the application has subscribed for is always sent up. Example: If the application has subscribed for ACL connection events it will receive a CSR_BT_CM_ACL_CONNECT_IND message even if the creation of an ACL connection fails.
CSR_BT_CM_EVENT_MASK_COND_SUCCESS	The extended information that the application has subscribed for is only sent up if status is success. Example: If the application has subscribed for ACL connection events the application will only receive a CSR_BT_CM_ACL_CONNECT_IND message if an ACL has been established with success.

4.44 CSR_BT_CM_SYNC

Primitives \ Parameters	type	syncHandle	deviceAddr	incoming	packetType	maxLatency	reTxEffort	rxBdw	txBdw	linkType	txInterval	weSco	reservedSlots	txPacketLength	airMode	voiceSettings	reason	resultCode	resultSupplier
CSR_BT_CM_SYNC_CONNECT_IND	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓
CSR_BT_CM_SYNC_RENEGOTIATE_IND	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓
CSR_BT_CM_SYNC_DISCONNECT_IND	✓	✓	✓														✓	✓	✓
CSR_BT_CM_EXT_SYNC_CONNECT_IND	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓

Table 44: CSR_BT_CM_SYNC Primitives

Description

If the application has subscribed for Synchronous Connection Events by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_SYNCHRONOUS_CONNECTION, it will receive the CSR_BT_CM_SYNC_CONNECT_IND message whenever a new synchronous connection has been or has attempted to be established, the CSR_BT_CM_SYNC_RENEGOTIATE_IND message whenever an existing synchronous connection has been or attempted to be reconfigured and the CSR_BT_CM_SYNC_DISCONNECT_IND message whenever a synchronous connection has been or attempted to be released.
If the application has subscribed for the Extended Synchronous Connection Events by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_EXT_SYNC_CONNECTION, it will receive the CSR_BT_CM_EXT_SYNC_CONNECT_IND message when the CM has read the extra audio parameters wEsco and txInterval from the Bluecore chip.

Parameters

type — Signal identity, CSR_BT_CM_SYNC_CONNECT_IND/RENEGOTIATE_IND/DISCONNECT_IND

syncHandle — Connection handle to be used for identifying the synchronous connection that has been connected, changed or disconnected.

deviceAddr — The Bluetooth® device address of the remote Bluetooth® device.

incoming — Boolean telling if the new synchronous connection is initiated by the local host (FALSE) or a remote host (TRUE).

packetType — The packetType parameter is a bit field defined as:

Value	Parameter description
0x0001	HV1 may be used
0x0002	HV2 may be used
0x0004	HV3 may be used
0x0008	EV3 may be used
0x0010	EV4 may be used
0x0020	EV5 may be used

0x0040	2-EV3 may not be used
0x0080	3-EV3 may not be used
0x0100	2-EV5 may not be used
0x0200	3-EV5 may not be used
0x0400	Reserved for future use
0x0800	Reserved for future use
0x1000	Reserved for future use
0x2000	Reserved for future use
0x4000	Reserved for future use
0x8000	Reserved for future use

Examples of packetType values that might be useful are calculated below:

Packet	PacketType	Calculation
HV1	0x03C1	0x0001 + 0x0040 + 0x0080 + 0x0100 + 0x0200
HV2	0x03C2	0x0002 + 0x0040 + 0x0080 + 0x0100 + 0x0200
HV3	0x03C4	0x0004 + 0x0040 + 0x0080 + 0x0100 + 0x0200
EV3	0x03C8	0x0008 + 0x0040 + 0x0080 + 0x0100 + 0x0200
EV4	0x03D0	0x0010 + 0x0040 + 0x0080 + 0x0100 + 0x0200
EV5	0x03E0	0x0020 + 0x0040 + 0x0080 + 0x0100 + 0x0200
2-EV3	0x0380	0x0080 + 0x0100 + 0x0200
3-EV3	0x0340	0x0040 + 0x0100 + 0x0200
2-EV5	0x02C0	0x0080 + 0x0040 + 0x0200
3-EV5	0x01C0	0x0080 + 0x0040 + 0x0100

maxLatency

Value	Parameter description
0x0000-0x0003	Reserved
0x0004-0xFFFE	This is a value in milliseconds representing the upper limit of the sum of the synchronous interval, the size of the eSCO window.
0xFFFF	Do not care

reTxEffort

Value	Parameter description
0x00	No retransmissions
0x01	At least one retransmission, optimized for power consumption
0x02	At least one retransmission, optimized for link quality.
0xFF	Do not care
0x03-0xFE	Reserved

rxBdw

If the parameter ‘incoming’ is set to FALSE then rxBandWidth is: Receive bandwidth in octets per second.

If ‘incoming’ is set to TRUE then rxBandWidth is:

Value	Parameter description
0x00000000-0xFFFFFFFE	Maximum received bandwidth in octets per second
0xFFFFFFFF	Do not care

txBdw

If the parameter ‘incoming’ is set FALSE then txBandWidth is: Transmit bandwidth in octets per second.

If ‘incoming’ is set to TRUE then txBandWidth is:
Transmit bandwidth in octets per second

Value	Parameter description
0x00000000-0xFFFFFFFE	Maximum possible transmit bandwidth in octets per second
0xFFFFFFFF	Do not care

linkType

Value	Parameter description
0x00	SCO connection
0x01	Reserved
0x02	eSCO connection
0x03-0xFF	Reserved

txInterval — Time between two consecutive SCO/eSCO instants measured in slots.

weSco — The size of the retransmission window measured in slots.

reservedSlots — Number of reserved slots within a txInterval

rxPacketLength — Length in bytes of the eSCO payload in the receive direction.
Please note that in the CSR_BT_CM_SYNC_RENEGOTIATE_IND message is this parameter reserved for future use.

txPacketLength — Length in bytes of the eSCO payload in the transmit direction.
Please note that in the CSR_BT_CM_SYNC_RENEGOTIATE_IND message is this parameter reserved for future use.

airMode

Value	Parameter description
0x00	µ-law log
0x01	A-law log
0x02	CVSD
0x03	Transparent Data
0x04-0xFF	Reserved

voiceSettings — The voice setting parameters used when creating or accepting an incoming synchronous connection. For more information refers to [BT].

Value	Parameter desc iption
00XXXXXXXX	Input Coding: Linear
01XXXXXXXX	Input Coding: µ-law Input Coding
10XXXXXXXX	Input Coding: A-law Input Coding
11XXXXXXXX	Reserved for future use
XX00XXXXXX	Input Data Format: 1's complement
XX01XXXXXX	Input Data Format: 2's compleme t
XX10XXXXXX	Input Data Format: Sign-Magnitude
XX11XXXXXX	Input Data Format: Unsigned
XXXX0XXXXX	Input Sample Size: 8-bit (Only for linear PCM)
XXXX1XXXXX	Input Sample Size: 16-bit (Only for linear PCM)
XXXXXXnnnX	Linear PCM Bit Position: number of bits positions that MSB of sample is away from starting at MSB (Only for linear PCM)
XXXXXXXX00	Air Coding Format: CVSD
XXXXXXXX01	Air Coding Format: µ-law
XXXXXXXX10	Air Coding Format: A-law
XXXXXXXX11	Air Coding Format: Transparent Data

status — The HCI status. Possible values for this parameter are defined in the HCI error codes section of [BT].

Please note that if the condition parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message is set to CSR_BT_CM_EVENT_MASK_COND_SUCCESS then the application will only receive these events if the synchronous connection has been established, renegotiated or released with success.

reason — The HCI reason why the synchronous connection is disconnected. Possible values for

	this parameter are defined in the HCI error codes section of [BT].
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.45 CSR_BT_CM_MODE_CHANGE and CSR_BT_CM_SNIFF_SUB_RATING

Primitives \ Parameters	type	deviceAddr	btConnId	mode	length	maxTxLatency	maxRxLatency	minRemoteTimeout	minLocalTmeout	resultCode	resultSupplier
CSR_BT_CM_MODE_CHANGE_IND	✓		✓	✓	✓					✓	✓
CSR_BT_CM_SNIFF_SUB_RATING_IND	✓	✓				✓	✓	✓	✓	✓	✓

Table 45: CSR_BT_CM_MODE_CHANGE and CSR_BT_CM_SNIFF_SUB_RATING Primitives

Description

If the application has subscribed for Mode Change and Sniff Sub Rating Events by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_MODE_CHANGE, it will receive the CSR_BT_CM_MODE_CHANGE_IND message whenever the connection associated with the device address changes or attempt to change between active mode, sniff mode and park mode, and the CSR_BT_CM_SNIFF_SUB_RATING_IND message whenever the connection associated with the device address has either enabled or attempted to enable sniff subrating or the sniff subrating parameters have been or attempted to be renegotiated.

Parameters

type — Signal identity, CSR_BT_CM_MODE_CHANGE_IND/CSR_BT_CM_SNIFF_SUB_RATING_IND.

deviceAddr — The Bluetooth address of the peer device.

btConnId — The BT connection ID.

Mode — The current link mode, where the mode parameter values are defined in csr_bt_cm_prim.h.

Value	Parameter description
ACTIVE_MODE	Indicates that the link is in active mode
SNIFF_MODE	Indicates that the link is in sniff mode
PARK_MODE	Indicates that the link is in park mode

length

If the current mode is active mode this parameter is not relevant, i.e. ignore it.

If the current mode is sniff mode this parameter indicates the number of baseband slots between sniff intervals.

Time between sniff intervals = 0.625 msec (1 Baseband slot)
Range for N: 0x0002-0xFFFE
Time Range: 1.25 msec-40.9 sec

If the current mode is park mode this parameter indicates the number of baseband slots between consecutive beacons.

Interval length = N * 0.625 msec (1 baseband slot)
Range for N: 0x0002-0xFFFE
Time range: 1.25 msec-40.9 seconds

maxTxLatency — Maximum latency for data being transmitted from the local device to the remote device.

	Latency = N * 0.625 msec (1 baseband slot) Range for N: 0x0000 – 0xFFFE Time range: 0 sec - 40.9 sec
maxRxLatency	Maximum latency for data being received by the local device from the remote device. Latency = N * 0.625 msec (1 baseband slot) Range for N: 0x0000 – 0xFFFE Time range: 0 sec - 40.9 sec
minRemoteTimeout	The base sniff subrate timeout in baseband slots that the remote device shall use. Timeout = N * 0.625 msec (1 baseband slot) Range for N: 0x0000 – 0x8000 Time range: 0 sec – 20.5 sec
minLocalTimeout	The base sniff subrate timeout in baseband slots that the remote device shall use. Timeout = N * 0.625 msec (1 baseband slot) Range for N: 0x0000 – 0x8000 Time range: 0 sec – 20.5 sec
status	The HCI status. Possible values for this parameter are defined in the HCI error codes section of [BT]. Please note that if the condition parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message is set to CSR_BT_CM_EVENT_MASK_COND_SUCCESS then the application will only receive these events if the mode actually has changed between active, sniff or park, or if sniff subrating has been enabled or the sniff subrating parameters have been renegotiated.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.46 CSR_BT_CM_ROLE_CHANGE

Primitives \ Parameters	type	deviceAddr	role	resultCode	resultSupplier
CSR_BT_CM_ROLE_CHANGE_IND	✓	✓	✓	✓	✓

Table 46: CSR_BT_CM_ROLE_CHANGE Primitives

Description

If the application has subscribed for Role Change Event by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_ROLE_CHANGE, it will receive the CSR_BT_CM_ROLE_CHANGE_IND message whenever the role of the connection associated with the device address have been or has attempt to be changed.

Parameters

type — Signal identity, CSR_BT_CM_ROLE_CHANGE_IND.

deviceAddr — The Bluetooth address of the peer device.

role — The role of the connection associated with the device address. Values are defined in csr_bt_profiles.h.

Value	Parameter description
MASTER_ROLE	Currently the master for the specified device address.
SLAVE_ROLE	Currently the slave for the specified device address.

resultCode — The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.

resultSupplier — This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.47 CSR_BT_CM_LSTO_CHANGE

Primitives \ Parameters	type	deviceAddr	timeout
CSR_BT_CM_LSTO_CHANGE_IND	✓	✓	✓

Table 47: CSR_BT_CM_LSTO_CHANGE Primitives

Description

If the application has subscribed for Link Supervision Timeout Changed Event by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_LSTO_CHANGE, it will receive the CSR_BT_CM_LSTO_CHANGE_IND message whenever the Link Supervision Timeout timer of the connection associated with the device address has changed. Please note that only the Master of the connection can change the value of the Link Supervision Timer. Also note, that if the peer device is the Master of the connection and it changes the Link Supervision Timer the CSR_BT_CM_LSTO_CHANGE_IND will only be sent to the application if and only if the local and the remote device support Bluetooth version 2.1 or higher.

Parameters

type — Signal identity, CSR_BT_CM_LSTO_CHANGE_IND.

deviceAddr — The Bluetooth address of the peer device.

timeout — The link supervision timeout value of the connection associated with the device address.

Value	Parameter description	
0xXXXX	No Link Supervision Timeout	0x0000
	Range	0x0001 to 0xFFFF
	Default	0x7D
	Mandatory Range	0x0190 to 0xFFFF
	Time	timeout * 0.625 msec
	Time Range	0.625 msec to 40.9 sec
	Time Default	20 sec

4.48 CSR_BT_CM_BLUECORE_INITIALIZED

Primitives \ Parameters	type
CSR_BT_CM_BLUECORE_INITIALIZED_IND	✓

Table 48: CSR_BT_CM_BLUECORE_INITIALIZED Primitive

Description

If the application has subscribed for a Bluecore Initialized event by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_BLUECORE_INITIALIZED, it will receive the CSR_BT_CM_BLUECORE_INITIALIZED_IND when BlueCore is initialized. Please note the CSR_BT_CM_SET_EVENT_MASK_REQ message can be sent momentarily from the application

Parameters

type Signal identity, CSR_BT_CM_BLUECORE_INITIALIZED_IND.

4.49 CSR_BT_CM_ACL_CONNECTION

Parameters / Primitives	type	deviceAddr	incoming	cod	aclConnHandle	resultCode	resultSupplier
CSR_BT_CM_ACL_CONNECT_IND	✓	✓	✓	✓	✓	✓	✓
CSR_BT_CM_ACL_DISCONNECT_IND	✓	✓				✓	✓

Table 49: CSR_BT_CM_ACL_CONNECT Primitives

Description

If the application has subscribed for ACL connection events by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_ACL_CONNECTION, it will receive the CSR_BT_CM_ACL_CONNECT_IND message whenever ACL connection has been or attempted to be created, and the CSR_BT_CM_ACL_DISCONNECT_IND message whenever the ACL connection is released.

Parameters

type	Signal identity, CSR_BT_CM_ACL_CONNECT_IND/ CSR_BT_CM_ACL_DISCONNECT_IND.
deviceAddr	The Bluetooth address of the peer device.
incoming	Boolean telling if the new ACL connection is initiated by the local host (FALSE) or a remote host (TRUE).
cod	The class of device of the remote device. Please note that this parameter is only valid on incoming connections meaning when the “Incoming” parameter is set to TRUE.
aclConnHandle	ACL connection handle
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.50 CSR_BT_CM_ACL_DETACH

Primitives \ Parameters	type	phandle	deviceAddr	flags	resultCode	resultSupplier
CSR_BT_CM_ACL_DETACH_REQ	✓	✓	✓	✓		
CSR_BT_CM_ACL_DETACH_CFM	✓				✓	✓

Table 50: CSR_BT_CM_READ_LOCAL_VERSION Primitives

Description

This signal can be used for detaching already established ACLs from any higher layer tasks at any given time they prefer.

The signal can be used in two different modes it can either detach a specific ACL or it can detach all established ACLs. If used for detaching all established ACLs, then only the already established ACLs, at the time the request is received in the CM, will be disconnected. Meaning, if other devices setup a new ACL after the CM has received the request for detaching all ACLs then this new ACL will not be detached as a consequence of the previous request.

Since this signal can be used for detaching one or all ACLs the CSR_BT_CM_ACL_DISCONNECT_IND, described in section 4.49 will be sent to the calling process, one or several times in between the CSR_BT_CM_ACL_DETACH_REQ and corresponding CFM, depending on the Bluetooth address specified in deviceAddr in the request.

The function:

CsrBtCmAclDetachReqSend (CsrSchedQid thePhandle, deviceAddr_t theDeviceAddr, CsrUint32 flags);

defined in csr_bt_cm_lib.h, builds and sends the CSR_BT_CM_ACL_DETACH_REQ primitive to the Connection Manager.

Parameters

type — Signal identity, CSR_BT_CM_ACL_DETACH_REQ/CFM.

phandle — The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.

deviceAddr — The Bluetooth address of the ACL which should be detached. If the Bluetooth address is set to zero, all established ACLs will be detached.

flags — Reports the result of detach procedure. The result values are defined in csr_bt_profiles.h and can be:

CSR_BT_CM_ACL_DETACH_ALWAYS: Always detach.

CSR_BT_CM_ACL_DETACH_EXCLUDE_L2CAP: Do not detach if L2CAP connections exists on the given ACL.

CSR_BT_CM_ACL_DETACH_EXCLUDE_RFC: Do not detach if RFC connections exists on the given ACL.

CSR_BT_CM_ACL_DETACH_EXCLUDE_BNEP: Do not detach if BNEP connections exists on the given ACL.

	CSR_BT_CM_ACL_DETACH_EXCLUDE_LE: Do not detach if BR/EDR based GATT (low energy) connections exists on the given ACL. **CSR_BT_CM_ACL_DETACH_EXCLUDE_ALL**: Do not detach if any logical connections exists on the given ACL.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.51 CSR_BT_CM_READ_FAILED_CONTACT_COUNTER

Primitives \ Parameters	type	phandle	deviceAddr	resultCode	resultSupplier	failedContactCount
CSR_BT_CM_READ_FAILED_CONTACT_COUNTER_REQ	✓	✓	✓			
CSR_BT_CM_READ_FAILED_CONTACT_COUNTER_CFM	✓		✓	✓	✓	✓

Table 51: CSR_BT_CM_READ_FAILED_CONTACT_COUNTER Primitives

Description

This signal can be used for reading the failed contact counter for a particular device. This can be used for monitoring the quality of the link.

The function:

CsrBtCmReadFailedContactCounterReqSend (CsrSchedQid phandle, deviceAddr_t deviceAddr);

defined in csr_bt_cm_lib.h, builds and sends the CSR_BT_CM_READ_FAILED_CONTACT_COUNTER_REQ primitive to the Connection Manager.

Parameters

type	Signal identity, CSR_BT_CM_READ_FAILED_CONTACT_COUNTER _REQ/CFM.
phandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.
deviceAddr	The Bluetooth address of the ACL whose failed contact counter should be read.
failedContactCount	The value of the failed contact counter.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.52 CSR_BT_CM_SWITCH_ROLE

Primitives \ Parameters	type	appHandle	deviceAddr	resultCode	resultSupplier	role	roleType	config
CSR_BT_CM_SWITCH_ROLE_REQ	✓	✓	✓			✓	✓	✓
CSR_BT_CM_SWITCH_ROLE _CFM	✓		✓	✓	✓	✓	✓	

Table 52: CSR_BT_CM_SWITCH_ROLE Primitives

Description

This signal, CSR_BT_CM_SWITCH_ROLE_REQ, can be used for changing the role of the given ACL connection. When the role switch has occurred (or the request has failed), the application will receive a CSR_BT_CM_SWITCH_ROLE_CFM containing the result. The CM will not interfere with the given ACL connection (e.g. perform role switch) until it has been released by the application. This can be achieved by using **UNDEFINED_ROLE** as *role* and **CSR_BT_CM_SWITCH_ROLE_TYPE_INVALID** as *roleType*.

Note: The peer device might change the role of the given ACL connection. If the application needs to be aware of this, the event subscription feature described in Section 4.43, should be used.

Note: It is the application's responsibility to ensure that the given ACL connection is not in low power or the request will fail. To ensure this, the application can use the CSR_BT_CM_MODE_CHANGE_REQ signal (see Section 4.54).

The function:

CsrBtCmSwitchRoleReqSend (CsrSchedQid phandle, deviceAddr_t deviceAddr, CsrUint8 role,

cm_role_type_t roleType, CsrUint32 config);

defined in csr_bt_cm_lib.h, builds and sends the CSR_BT_CM_SWITCH_ROLE_REQ primitive to the Connection Manager.

Parameters

type	Signal identity, CSR_BT_CM_SWITCH_ROLE_REQ/CFM.
phandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.
deviceAddr	The Bluetooth address of the ACL whose failed contact counter should be read.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
role	The role of the connection associated with the device address. Values are defined in csr_bt_cm_prim.h.

Value	Parameter description
MASTER_ROLE	The local device is/should become master.
SLAVE_ROLE	The local device is/should become slave.
UNDEFINED_ROLE	Role could be not determined.

Note: The **UNDEFINED_ROLE** shall be used for releasing control of the ACL connection.

roleType

Configuration parameter for the switch role request. The values are defined in csr_bt_cm_prim.h and can be:

CSR_BT_CM_SWITCH_ROLE_TYPE_INVALID: Application wishes to release control over the ACL connection. The CM is now free to change the role according to its own internal rules.

CSR_BT_CM_SWITCH_ROLE_TYPE_ONESHOT: Application wishes to switch role and gain control of the ACL connection. The CM will no longer switch role on the given connection autonomously.

Config

Reserved for future use. Shall be zero.

4.53 CSR_BT_CM_MODE_CHANGE_CONFIG

Primitives \ Parameters	type	phandle	deviceAddr	config	resultCode	resultSupplier
CSR_BT_CM_MODE_CHANGE_CONFIG_REQ	✓	✓	✓	✓		
CSR_BT_CM_MODE_CHANGE_CONFIG_CFM	✓		✓		✓	✓

Table 53: CSR_BT_CM_MODE_CHANGE_CONFIG Primitives

Description

The application can decide if CSR Synergy Bluetooth or the application itself is handling the low power management on a single or all ACL connections, by sending a CSR_BT_CM_MODE_CHANGE_CONFIG_REQ message. By default CSR Synergy Bluetooth is responsible for dealing with the low power management.

Note that even if the application is responsible for controlling the low power mode, the peer device can still change it, and CSR Synergy Bluetooth will set the ACL connection in ACTIVE mode if an ACL connection is placed in PARK mode or if CSR Synergy Bluetooth shall establish or release a logical connection, which is or will be attached to an ACL connection that currently are in Sniff mode.

The function:

CsrBtCmModeChangeConfigReqSend (CsrSchedQid phandle, deviceAddr_t deviceAddr, CsrUint32 config);

defined in csr_bt_cm_lib.h, builds and sends the CSR_BT_CM_MODE_CHANGE_CONFIG_REQ primitive to the Connection Manager.

Parameters

type — Signal identity, CSR_BT_CM_MODE_CHANGE_CONFIG_REQ/CFM.

phandle — The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.

deviceAddr — The address of the connected device for which low power mode should be configured. If the address is set to 0000:00:000000 the configuration will take place for all existing and future connections.

config — The following values are allowed (see csr_bt_cm_prim.h):

- **CSR_BT_CM_MODE_CHANGE_DISABLE**: The default behavior. CSR Synergy Bluetooth is controlling low power handling
- **CSR_BT_CM_MODE_CHANGE_ENABLE**: The Application is controlling low power handling

resultCode — The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.

resultSupplier — This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.54 CSR_BT_CM_MODE_CHANGE

Primitives \ Parameters	type	phandle	deviceAddr	mode	forceSniffSettings	sniffSettings	resultCode	resultSupplier	interval	taskResult	task
CSR_BT_CM_MODE_CHANGE_REQ	✓	✓	✓	✓	✓	✓					
CSR_BT_CM_MODE_CHANGE_CFM	✓		✓	✓			✓	✓	✓	✓	✓

Table 54: CSR_BT_CM_MODE_CHANGE Primitives

Description

This signal, CSR_BT_CM_MODE_CHANGE_REQ, can be used for changing the power mode of the given ACL connection (E.g. this command allowed the application to change between Active Mode and Sniff Mode). When the mode is changed (or the request has failed), the application will receive a CSR_BT_CM_MODE_CHANGE_CFM containing the result. Please note that if the application want to know whenever the mode of the ACL connections changes it must subscribe for a mode change event. How this can be done in describe in section 4.43.

The functions:

CsrBtCmSniffModeReqSend(CsrSchedQid phandle, deviceAddr_t deviceAddr, CsrUint16 maxInterval, CsrUint16 minInterval, CsrUint16 attempt, CsrUint16 timeout, CsrBool forceSniffSettings);

and

CsrBtCmExitSniffModeReqSend (CsrSchedQid phandle, deviceAddr_t deviceAddr);

defined in csr_bt_cm_lib.h, builds and sends the CSR_BT_CM_MODE_CHANGE_REQ primitive to the Connection Manager. The function *CsrBtCmSniffModeReqSend* must be use when the ACL connection indentified by the deviceAddr must try to enter sniff mode, and the function *CsrBtCmExitSniffModeReqSend* must be use when the ACL connection must exit sniff mode.

Parameters

type — Signal identity, CSR_BT_CM_MODE_CHANGE_REQ/CFM.

phandle — The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.

deviceAddr — The Bluetooth address of the peer device which identifies the ACL for which a mode change is requested

mode — The requested link mode, where the mode parameter values which must be used are defined in csr_bt_cm_prim.h.

Value	Parameter description
ACTIVE_MODE	Indicates that the link is in active mode
SNIFF_MODE	Indicates that the link is in sniff mode

forceSniffSettings — If 'forceSniffSettings' is set to TRUE and the current mode of the ACL connection is Sniff then will the CM Exit Sniff mode and enter sniff mode again with the given sniff settings. If FALSE and current mode is sniff, it will just stay in sniff mode.

sniffSettings

The structure of the *sniffSettings* parameter is defined in csr_bt_dm_prim.h as:

```
typedef struct
{
  CsrUint16  max_interval;
  CsrUint16     min_interval;
  CsrUint16   attempt;
  CsrUint16  timeout;
} SNIFF_SETTINGS_T;
```

where, the *max_interval* and *min_interval* parameters specify the requested acceptable maximum and minimum periods in the Sniff Mode. The *min_interval* must be larger, the *max_interval* and the *max_interval* must be less than the Link Supervision Timeout configuration. The sniff interval defines the amount of time between each consecutive sniff period. The range is: 0x0002 to 0xFFFE (a time range from 1.25 msec to 40.9 sec). Please note that **only** even values are valid. The mandatory range is: 0x0006 to 0x0540.

The *attempt* parameter defines the number of Baseband received slots for sniff attempts and the *timeout* parameter defines the number of Baseband received slots for sniff timeout. The range for the *attempt* parameter is: 0x0001 to 0x7FFF and the range for the *timeout* parameter is 0x0000 to 0x7FFF. For more information about the *attempt* and *timeout* parameters, please refer to [BT].

interval

If the current mode is active mode this parameter is not relevant, i.e. ignore it.

If the current mode is Sniff mode this parameter indicates the number of baseband slots between sniff intervals.

Time between sniff intervals = 0.625 msec (1 Baseband slot)
Range for N: 0x0002-0xFFFE
Time Range: 1.25 msec-40.9 sec

If the current mode is park mode this parameter indicates the number of baseband slots between consecutive beacons.
Interval length = N * 0.625 msec (1 baseband slot)
Range for N: 0x0002-0xFFFE
Time range: 1.25 msec-40.9 seconds

resultCode

The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.

resultSupplier

This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

taskResult

This parameter can be used for determined the error code from the protocol layer defined in ‘task’.

If ‘task’ is set to CSR_BT_TASK_CM the value of ‘result’ and ‘taskResult’ match each other.

If ‘task’ is set to CSR_BT_TASK_DM the ‘taskResult’ are defined in csr_bt_hci.h, which match the error, codes defined in [BT].

task

If ‘taskResult’ is valid, this parameter indicates which task the error originated from. This parameter can assume the following values (found in csr_bt_profiles.h):

- CSR_BT_TASK_CM

- CSR_BT_TASK_DM

4.55 CSR_BT_CM_LOGICAL_CHANNEL_TYPE

Primitives \ Parameters	type	logicalChannelTypeMask	deviceAddr	numberOfGuaranteedLogicalChannels
CSR_BT_CM_LOGICAL_CHANNEL_TYPES_IND	✓	✓	✓	✓

Table 55: CSR_BT_CM_LOGICAL_CHANNEL_TYPE Primitives

Description

The CM keeps track of the number and type of connections active at a given time and indicates it to the application via the CSR_BT_CM_LOGICAL_CHANNEL_TYPES_IND.

Note, the CM issues the CSR_BT_CM_LOGICAL_CHANNEL_TYPES_IND only if the application has subscribed for this event, see section 4.43.

Parameters

type — Signal identity, CSR_BT_CM_LOGICAL_CHANNEL_TYPES_IND.

deviceAddr — The Bluetooth address of the peer device which identifies the ACL for which a mode change is requested

logicalChannelTypeMask — 32-bit bitmask to indicate the type of connection established if any. The values allowed are defined in csr_bt_cm_prim.h.

Value	Parameter description
CSR_BT_NO_ACTIVE_LOGICAL_CHANNEL	0x00000000. Indicates that there are no logical channels.
CSR_BT_ACTIVE_DATA_CHANNEL	0x00000001. Indicates that there is t least one active data channel
CSR_BT_ACTIVE_CONTROL_CHANNEL	0x00000002. Indicates that there is at least one control channel active
CSR_BT_ACTIVE_STREAM_CHANNEL	0x00000004. Indicates that at least one AV stream is active

numberOfGuaranteedLogicalChannels — This field indicates how many AV active streams exist at a given time.

4.56 CSR_BT_CM_READ_REMOTE_FEATURES

Primitives \ Parameters	type	deviceAddr	remoteLmpFeatures[8]	resultCode	resultSupplier
CSR_BT_CM_READ_REMOTE_ FEATURES_IND	✓	✓	✓	✓	✓

Table 56: CSR_BT_CM_READ_REMOTE_ FEATURES Primitives

Description

If the application has subscribed for Read Remote Features Event by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_REMOTE_FEATURES, it will receive the CSR_BT_CM_READ_REMOTE_ FEATURES_IND message whenever the Remote Features has been read.

Parameters

type	Signal identity, CSR_BT_CM_READ_REMOTE_ FEATURES_IND.
deviceAddr	The device address of the device, which remote features has been read.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
remoteLmpFeatures[8]	Returns a bit mask of the features the remote device supports. See [BT].

4.57 CSR_BT_CM_A2DP_BIT_RATE

Primitives \ Parameters	type	deviceAddr	streamIdx	bitRate
CSR_BT_CM_A2DP_BIT_RATE_IND	✓	✓	✓	✓

Table 57: CSR_BT_CM_A2DP_BIT_RATE Primitives

Description

If the application has subscribed for the A2DP bit rate Event by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_A2DP_BIT_RATE, it will receive the CSR_BT_CM_A2DP_BIT_RATE_IND message whenever the bit rate is indicated by the AV profile if it is running and an A2DP stream is active.

Parameters

type	Signal identity, CSR_BT_CM_A2DP_BIT_RATE_IND.
deviceAddr	The device address of the device connected to.
streamIdx	Identifier of the stream, the bit rate indicated applies to. This is needed in case more than one stream exist at a time.
bitRate	32-bit value with the bit rate used for the streaming connection if that value is available. Otherwise, it will be: - CSR_BT_A2DP_BIT_RATE_UNKNOWN (0x00000000). - CSR_BT_A2DP_BIT_RATE_STREAM_SUSPENDED (0xFFFFFFFE) - CSR_BT_A2DP_BIT_RATE_STREAM_DISCONNECTED (0xFFFFFFFF)

4.58 CSR_BT_CM_INQUIRY_PAGE_EVENT

Primitives \ Parameters	type	inquiry	paging
CSR_BT_CM_INQUIRY_PAGE_EVENT_IND	✓	✓	✓

Table 58: CSR_BT_CM_INQUIRY_PAGE_EVENT Primitives

Description

If the application has subscribed for inquiry and Page Event by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_INQUIRY_PAGE_STATE, it will receive the CSR_BT_CM_INQUIRY_PAGE_EVENT_IND message whenever an inquiry operation is either started or stopped and when a paging operation (outgoing connection) is started or stopped.

Parameters

type	Signal identity, CSR_BT_CM_INQUIRY_PAGE_EVENT_IND.	
Inquiry	One of the following values:	
	CSR_BT_CM_INQUIRY_TYPE_START	0x00 → meaning inquiry operation ongoing
	CSR_BT_CM_INQUIRY_TYPE_STOP	0x01 → meaning no inquiry ongoing
paging	One of the following values:	
	CSR_BT_CM_PAGE_TYPE_START	0x00 → meaning paging ongoing
	CSR_BT_CM_PAGE_TYPE_STOP	0x01 → meaning no paging ongoing

4.59 CSR_BT_CM_BLE_CONNECTION

Primitives \ Parameters	type	deviceAddr	bleUsed	bleTiming	resultCode	resultSupplier
CSR_BT_CM_BLE_CONNECTION_IND	✓	✓	✓	✓	✓	✓

Table 59: CSR_BT_CM_BLE_CONNECTION Primitives

Description

If the application has subscribed for BLE Connection Event by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_BLE_CONNECTION, it will receive the CSR_BT_CM_BLE_CONNECTION_IND message whenever a BLE connection is established or it is tried and fails.

Parameters

type	Signal identity, CSR_BT_CM_BLE_CONNECTION_IND.
deviceAddr	The device address of the device connected to.
bleUsed	Boolean to tell whether BLE actually is used for the connection
bleTiming	16-bit value with the timing interval used for the BLE connection
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.60 CSR_BT_CM_ENCRYPT_CHANGE

Primitives \ Parameters	type	deviceAddr	encrypted	resultCode	resultSupplier
CSR_BT_CM_ENCRYPT_CHANGE_IND	✓	✓	✓	✓	✓

Table 60: CSR_BT_CM_ENCRYPT_CHANGE Primitives

Description

If the application has subscribed for a Encryption Change Event by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_ENCRYPT_CHANGE, it will receive the CSR_BT_CM_ENCRYPT_CHANGE_IND message whenever Link Level Encryption is changed

Parameters

type	Signal identity, CSR_BT_CM_ENCRYPT_CHANGE_IND.
deviceAddr	The Bluetooth address of the peer device which identifies the ACL for which the link layer encryption has been enabled/disabled
encrypted	Boolean that indicates whether the ACL link is encrypted or not. If TRUE the link level encryption is ON.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.61 CSR_BT_CM_ ALWAYS_MASTER_DEVICES

Primitives \ Parameters	type	phandle	deviceAddr	operation	resultCode	resultSupplier
CSR_BT_CM_ALWAYS_MASTER_DEVICES_REQ	✓	✓	✓	✓		
CSR_BT_CM_ALWAYS_MASTER_DEVICES_CFM	✓		✓		✓	✓

Table 61: CSR_BT_CM_ALWAYS_MASTER_DEVICES Primitives

Description

This message can be use to maintain list of remote devices for which the Device Manager will always try to become master during any ACL connection creation, even if there are no existing ACLs connected.
For locally-initiated connection requests to devices in the list, the Device Manager will prohibit role-switch, thus ensuring that the local device becomes master. For remotely-initiated connection requests to devices in the list, the Device Manager will request a role-switch during connection creation. This may or may not be accepted by the remote device. This primitive may be used for:

1) ADDING a new device to the list

2) DELETING a device from the list

3) CLEARING the entire list

Please note that this message should be used only to work around problems with severely misbehaving remote devices. Any other use is likely to produce a severely misbehaving local device and lead to major interoperability problems. E.g. this primitive should only be used when it is necessary to become master, even when there are no existing connections, because the remote device is badly behaved and will not role-switch after connection creation and it is likely that further ACLs will soon be connected.

Note this list is NOT stored in a non-volatile storage (NVS), e.g. if the stack is shutdown the application need to updated the list again.

The function:

CsrBtCmAlwaysMasterDevicesReqSend (CsrSchedQid phandle, deviceAddr_t deviceAddr, CsrUint16 operation);

defined in csr_bt_cm_lib.h, builds and sends the CSR_BT_CM_ALWAYS_MASTER_DEVICES_REQ primitive to the Connection Manager.

Parameters

type	Signal identity, CSR_BT_CM_ALWAYS_MASTER_DEVICES_REQ/CFM.
phandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.
deviceAddr	The Bluetooth address of the peer device, which shall be added to, or delete from, the list.
operation	Defines the operation, which must have one of the following values: **CSR_BT_CM_ALWAYS_MASTER_DEVICES_CLEAR**: CLEAR the entire list

	CSR_BT_CM_ALWAYS_MASTER_DEVICES_ADD: ADD a new device to the list **CSR_BT_CM_ALWAYS_MASTER_DEVICES_DELETE**: DELETE a device from the list Which are define in csr_bt_cm_prim.h
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.62 CSR_BT_CM_REGISTER

Primitives \ Parameters	type	phandle	context	serverChannel	resultCode	resultSupplier
CSR_BT_CM_REGISTER_REQ	✓	✓	✓	✓		
CSR_BT_CM_REGISTER_CFM	✓			✓	✓	✓

Table 62: CSR_BT_REGISTER Primitives

Description

An application that wishes to make use of a particular RFCOMM server channel may request it with the CSR_BT_REGISTER_REQ message. The CM will answer with the CSR_BT_CM_REGISTER_CFM. The operation will fail if the server channel requested is already in use. If the operation succeeds the result will take the value CSR_BT_RESULT_CODE_CM_SUCCESS and the result supplier will be CSR_BT_SUPPLIER_CM.

The function:

CsrBtCmPublicRegisterReqSend (CsrSchedQid phandle, CsrUint16 context, CsrUint8 serverChannel);

defined in csr_bt_cm_lib.h, builds and sends the CSR_BT_CM_REGISTER_REQ primitive to the Connection Manager.

Parameters

type	Signal identity, CSR_BT_CM_REGISTER_REQ/CFM.
phandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.
context	Not relevant for the application. Should take the value CSR_BT_CM_CONTEXT_UNUSED defined in csr_bt_cm_prim.h
serverChannel	8-bit value designating the server channel number desired. The value “CSR_BT_CM_SERVER_CHANNEL_DONT_CARE” defined in csr_bt_cm_prim.h shall be used if the application will accept whatever server channel the connection manager come up with.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.63 CSR_BT_CM_UNREGISTER

Primitives \ Parameters	type	serverChannel
CSR_BT_CM_UNREGISTER_REQ	✓	✓

Table 63: CSR_BT_UNREGISTER Primitives

Description

An application that has registered a server channel should free it again when it does no longer need it. To do that, the application shall issue the CSR_BT_UNREGISTER_REQ message. There is no confirmation as answer to that message. The operation will always succeed if the server channel given exists and is in use and otherwise the connection manager will ignore it.

The function:

CsrBtCmUnRegisterReqSend (CsrUint8 serverChannel);

defined in csr_bt_cm_lib.h, builds and sends the CSR_BT_CM_UNREGISTER_REQ primitive to the Connection Manager.

Parameters

type	Signal identity, CSR_BT_CM_UNREGISTER_REQ.
serverChannel	The server channel previously allocated

4.64 CSR_BT_CM_READ_ADVERTISING_CH_TX_POWER

Parameters / Primitives	type	appHandle	txPower	context	resultCode	resultSupplier
CSR_BT_CM_READ_ADVERTISING_CH_TX_POWER_REQ	✓	✓		✓		
CSR_BT_CM_READ_ADVERTISING_CH_TX_POWER_CFM	✓		✓		✓	✓

Table 64: CSR_BT_CM_READ_ADVERTISING_CH_TX_POWER Primitives

Description

Read TX power for low energy advertising channel.

Parameters

Type	Signal identity, CSR_BT_CM_READ_ADVERTISING_CH_TX_POWER_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
txPower	TX power of low energy advertising channel
Context	Opaque context number returned in CsrBtCmReadAdvertisingChTxPowerCfm
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

The functions:

CsrBtCmReadAdvertisingChTxPowerReqSend (CsrSchedQid appHandle, CsrUint16 context)

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_READ_ADVERTISING_CH_TX_POWER_REQ primitive to the Connection Manager.

4.65 CSR_BT_CM_LOCAL_NAME_CHANGE

Primitives \ Parameters	Type	localNAme
CSR_BT_CM_LOCAL_NAME_CHANGE_IND	✓	✓

Table 65: CSR_BT_CM_LOCAL_NAME_CHANGE Primitives

Description

If the application has subscribed for a Local Name Change Event by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_LOCAL_NAME_CHANGE, it will receive the CSR_BT_CM_LOCAL_NAME_CHANGE_IND message whenever Local Name is changed

Parameters

type	Signal identity, CSR_BT_CM_LOCAL_NAME_CHANGE_IND.
localName	Utf-8 string pointer containing the name of the local device.

4.66 CSR_BT_CM_LE_EVENT_ADVERTISING_IND

Primitives \ Parameters	type	event	advType	intervalMin	intervalMax	channelMap
CSR_BT_CM_LE_EVENT_ADVERTISING_IND	✓	✓	✓	✓	✓	✓

Table 66: CSR_BT_CM_LE_EVENT_ADVERTISING Primitives

Description

If the application has subscribed for Advertising events by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_LOW_ENERGY, it will receive the CSR_BT_CM_LE_EVENT_ADVERTISING_IND message whenever the advertising mode is changed.

Parameters

type	Signal identity, CSR_BT_CM_LE_EVENT_ADVERTISING_IND
event	State, use CSR_BT_CM_LE_MODE_... from csr_bt_cm_prim.h
advType	Type of advertising, use CSR_BT_CM_LE_ADVTYPE_.... from csr_bt_cm_prim.h
intervalMin	Minimum advertising interval (in slots, i.e. x * 0.625ms).
intervalMax	Maximum advertising interval (in slots, i.e. x * 0.625ms).
channelMap	Advertising channel map, use CSR_BT_CM_LE_CHANMAP_... from csr_bt_cm_prim.h

4.67 CSR_BT_CM_LE_EVENT_SCAN_IND

Primitives \ Parameters	type	event	scanType	interval	window
CSR_BT_CM_LE_EVENT_SCAN_IND	✓	✓	✓	✓	✓

Table 67: CSR_BT_CM_LE_EVENT_SCAN Primitives

Description

If the application has subscribed for Scan events by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_LOW_ENERGY, it will receive the CSR_BT_CM_LE_EVENT_SCAN_IND message whenever the scan mode is changed.

Parameters

type	Signal identity, CSR_BT_CM_LE_EVENT_SCAN_IND
event	State, use CSR_BT_CM_LE_MODE_... from csr_bt_cm_prim.h
scanType	Type of scanning, use CSR_BT_CM_LE_SCANTYPE_... from csr_bt_cm_prim.h
interval	Scan interval (in slots, i.e. x * 0.625ms).
window	Scan window (in slots, i.e. x * 0.625ms).

4.68 CSR_BT_CM_LE_EVENT_CONNECTION_IND

Primitives \ Parameters	type	event	deviceAddr	role	interval	timeout	latency	accuracy
CSR_BT_CM_LE_EVENT_CONNECTION_IND	✓	✓	✓	✓	✓	✓	✓	✓

Table 68: CSR_BT_CM_LE_EVENT_CONNECTION Primitives

Description

If the application has subscribed for LE Connection events by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_LOW_ENERGY, it will receive the CSR_BT_CM_LE_EVENT_CONNECTION_IND message whenever a LE connection is established / disconnected.

Parameters

type	Signal identity, CSR_BT_CM_LE_EVENT_ADVERTISING_IND
event	State, use CSR_BT_CM_LE_MODE_... from csr_bt_cm_prim.h
deviceAddr	The Bluetooth® address of the local device.
role	Role, use CSR_BT_ROLE_... from csr_bt_cm_prim.h
interval	Connection interval (in slots, i.e. x * 0.625ms).
timeout	Supervision timeout (in 10ms units).
latency	Connection latency (in slots, i.e. x * 0.625ms)
accuracy	Clock accurary, use CSR_BT_CM_LE_CLOCKACCU_... from csr_bt_cm_prim.h

4.69 CSR_BT_CM_HIGH_PRIORITY_DATA_IND

Primitives \ Parameters	type	event	deviceAddr	start
CSR_BT_CM_HIGH_PRIORITY_DATA_IND	✓	✓	✓	✓

Table 69: CSR_BT_CM_HIGH_PRIORITY_DATA_IND Primitives

Description

If the application has subscribed for a High Priority Data Event by setting the eventMask parameter in the CSR_BT_CM_SET_EVENT_MASK_REQ message to CSR_BT_CM_EVENT_MASK_SUBSCRIBE_HIGH_PRIORITY_DATA, it will receive the CSR_BT_CM_HIGH_PRIORITY_DATA_IND message whenever CM is starting to send high priority data or if it has stopped sending high priority data.

Parameters

type	Signal identity, CSR_BT_CM_HIGH_PRIORITY_DATA_IND
deviceAddr	The Bluetooth® address of the local device.
start	TRUE if high priority data has started been send, FALSE if it has stopped sending high priority data

4.70 CSR_BT_CM_LE_RECEIVER_TEST

Parameters / Primitives	type	appHandle	rxFrequency	resultCode	resultSupplier
CSR_BT_CM_LE_RECEIVER_TEST_REQ	✓	✓	✓		
CSR_BT_CM_LE_RECEIVER_TEST_CFM	✓			✓	✓

Table 70: CSR_BT_CM_LE_RECEIVER_TEST Primitives

Description

Initiate the Low Energy receiver test. To stop the Rx test and obtain the results, use CSR_BT_CM_LE_TEST_END_REQ/CFM.

Parameters

Type	Signal identity, CSR_BT_CM_LE_RECEIVER_TEST_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
rxFrequency	Rx frequency to perform the test on. Channel number is (2402-f)/2. Legal range is 0x00 to 0x27.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

The function:

CsrBtCmLeReceiverTestReqSend(CsrSchedQid appHandle, CsrUint8 rxFrequency)

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_LE_RECEIVER_TEST_REQ primitive to the CM.

4.71 CSR_BT_CM_LE_TRANSMITTER_TEST

Parameters / Primitives	type	appHandle	txFrequency	lengthOfTestData	packetPayload	resultCode	resultSupplier
CSR_BT_CM_LE_TRANSMITTER_TEST_REQ	✓	✓	✓	✓	✓		
CSR_BT_CM_LE_TRANSMITTER_TEST_CFM	✓					✓	✓

Table 71: CSR_BT_CM_LE_TRANSMITTER_TEST Primitives

Description

Initiate the Low Energy transmitter test. To stop the Tx test, use CSR_BT_CM_LE_TEST_END_REQ/CFM.

Parameters

type	Signal identity, CSR_BT_CM_LE_TRANSMITTER_TEST_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
txFrequency	Tx frequency to perform the test on. Channel number is (2402-f)/2. Legal range is 0x00 to 0x27.
lengthOfTestData	Size of the packets to transmit. Legal range is 0x00 to 0x25.
packetPayload	Payload pattern. Only values defined in the core specification are legal. See BT4.0 volume 2, part E, section 7.8.30 for details on the pattern.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

The function:

CsrBtCmLeTransmitterTestReqSend(CsrSchedQid appHandle, CsrUint8 txFrequency, CsrUint8 lengthOfTestData, CsrUint8 packetPayload)

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_LE_TRANSMITTER_TEST_REQ primitive to the CM.

4.72 CSR_BT_CM_LE_TEST_END

Parameters / Primitives	type	appHandle	numberOfPackets	resultCode	resultSupplier
CSR_BT_CM_LE_ TEST_END_REQ	✓	✓			
CSR_BT_CM_LE_ TEST_END_CFM	✓		✓	✓	✓

Table 72: CSR_BT_CM_LE_TEST_END Primitives

Description

Stop an ongoing Tx or Rx low energy radio test.

Parameters

Type	Signal identity, CSR_BT_CM_LE_TEST_END_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
numberOfPackets	Number of packets received for the Rx test. Value always 0 for the Tx tester.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

The function:

CsrBtCmLeTestEndReqSend(CsrSchedQid appHandle)

defined in csr_bt_cm_lib.h, build and send the CSR_BT_CM_LE_ TEST_END_REQ primitive to the CM.

5 Document References

Document	Reference
Bluetooth® Core Specification v.1.1, v.1.2, v.2.0 and v.2.1	[BT]
Bluetooth® Assigned Numbers	[BT12]

Terms and Definitions

BlueCore®	Group term for CSR's range of Bluetooth wireless technology chips
Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
CSR	Cambridge Silicon Radio
SIG	Special Interest Group
CM	Connection Manager
UniFi™	Group term for CSR's range of chips designed to meet IEEE 802.11 standards
VM	Virtual Machine

Document History

Revision	Date	History
1	26 SEP 11	Ready for release 18.2.0
2	12 DEC 11	Ready for release 18.2.1

TradeMarks, Patents and Licences

Unless otherwise stated, words and logos marked with ™ or ® are trademarks registered or owned by CSR plc or its affiliates. Bluetooth® and the Bluetooth logos are trademarks owned by Bluetooth SIG, Inc. and licensed to CSR. Other products, services and names used in this document may have been trademarked by their respective owners.

The publication of this information does not imply that any licence is granted under any patent or other rights owned by CSR plc.

CSR reserves the right to make technical changes to its products as part of its development programme.

While every care has been taken to ensure the accuracy of the contents of this document, CSR cannot accept responsibility for any errors.

Life Support Policy and Use in Safety-critical Compliance

CSR's products are not authorised for use in life-support or safety-critical applications. Use in such applications is done at the sole discretion of the customer. CSR will not warrant the use of its devices in such applications.

Performance and Conformance

Refer to www.csrsupport.com for compliance and conformance to standards information.