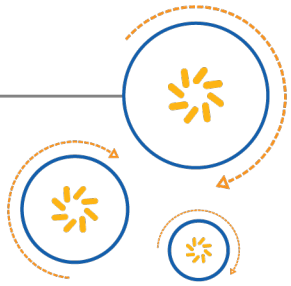
Qualcomm Technologies International, Ltd.



Operation of Bluetooth v2.1 Devices

Application Note

80-CT400-1 Rev. AK

October 19, 2017

Revision history

Revision	Date	Description
1	DEC 2007	Initial release. Alternative document number CS-00115441-TC.
2	MAR 2008	Bonding description updated.
3	APR 2008	Bonding description updated.
4	APR 2009	Technical update
5	JUL 2010	Minor editorial changes and updated to latest style guidelines.
6	JUL 2011	Firmware23d reference removed and updated to latest CSR style.
7	JAN 2012	Updated to latest CSR™ style.
8	APR 2014	Updated to latest CSR style.
9	SEP 2016	Updated to conform to QTI standards; no technical content was changed in this document revision.
AK	AUG 2017	Added to Content Management System. DRN updated to use Agile number. No technical content was changed in this document revision.

Contents

Tables

Figures

1 Initialization of Bluetooth devices

The Bluetooth version of a device is set up automatically, by the connection library, during initialization. If a device has Bluetooth v2.1 features enabled in the local supported features block (that is, PSKEY_LOCAL_SUPPORTED_FEATURES) then the connection library initializes as a Bluetooth v2.1 device and defaults into security mode 4.

If any Bluetooth v2.1 features are disabled in the local supported features block then the connection library initializes as a Bluetooth v2.0 device and defaults into security mode 2.

NOTE Unified 23 and later firmware has Bluetooth v2.1 features enabled by default and these should not be disabled. The connection library does not currently support Bluetooth v2.0 host with Bluetooth v2.1 controller.

A Bluetooth v2.1 device cannot leave security mode 4.

A Bluetooth v2.0 device cannot enter security mode 4. However, it may use any of the other security modes.

The Bluetooth version of a device cannot be changed after initialization.

2 Bluetooth v2.1 extended inquiry response

This feature of Bluetooth v2.1 allows extra information to be included in the inquiry response. The extended inquiry response (EIR) data must at least contain the device's local name, although if the device has no local name this may be of length zero or omitted altogether. If there is other data in the response; it is down to the application to ensure this is adhered to.

A device can be put into EIR inquiry mode using `ConnectionWriteInquiryMode` and the EIR data set up using `ConnectionWriteEirData`. The first byte of the EIR data should give the length of the first attribute, and the second byte the attribute type (see Table 2-1) the attribute data should then follow. The data section should then be followed by either the length of the next attribute or a null termination byte if it is the last section, (the null termination byte is not sent over the air).

The data types are:

Table 2-1 EIR data types

Hex Value	Attribute Type
`0x01`	Flags
`0x02/0x03`	Incomplete/Complete list of 16-bit UUIDs
`0x04/0x05`	Incomplete/Complete list of 32-bit UUIDs
`0x06/0x07`	Incomplete/Complete list of 128-bit UUIDs
`0x08/0x09`	Incomplete/Complete local name
`0x0a`	Inquiry Tx Power Level
`0x0b-0x0f`	OOB Data, see Table 2-2
`0xff`	Manufacturer Specific Data

The inquiry Tx power level can now be read/set by calling `ConnectionReadInquiryTxPowerLevel` and `ConnectionWriteInquiryTxPowerLevel`.

Inquiry can also be performed OOB if the device supports such a mechanism. OOB inquiry data should contain the sections described in Table 2-2:

Table 2-2 Mandatory and optional out of band inquiry data

Mandatory		Optional	
Hex Value	Description	Hex Value	Description
`0x0b`	Length of the optional data part	`0x0d`	COD
`0x0c`	BDADDR	`0x0e`	Simple Pairing `Hash C`
-	-	`0x0f`	Simple Pairing `Rand R`

Simple Pairing Hash C and Rand R should be obtained by calling `ConnectionSmReadLocalOobData` before the out of band inquiry takes place.

The `CL_DM_INQUIRY_RESULT_IND_T` structure has been expanded to hold EIR data if a device supporting EIR is discovered. It is up to the application to parse this data to get the attributes it needs.

3 Bluetooth v2.1 sniff sub-rating

Sniff sub-rating, rather than forcing devices to turn up at all sniff anchor points, allows devices to specify a maximum remote latency, that is, how often they need the remote device to be present when in sniff mode. This enables the device to skip some sniff anchor points to increase power saving. The remote latency is limited by the Link Manager (LM) to be less than the link supervision timeout.

On receiving ACL-C/ACL-U data, and when it initially enters sniff mode, a device with sub-rating enabled will stay in sniff mode for a negotiated timeout before it enters sub-rating mode. A device can specify its requirements for both the local and remote minimum timeouts, as well as the maximum remote latency using the `ConnectionSetSniffSubRatePolicy` function. A value of `N` for latency/timeout parameters gives a time of `N` x 0.625 ms, where `N` can be any value from `0x0000` to `0xfffe`. Setting all three values to `0x0000` is equivalent to disabling sub-rating; the LM defaults to these values if no others are specified by the application.

NOTE The maximum Bluetooth radio frequency hop rate of 1600 hops per second determines that time slot are set as multiples of 0.625 ms, see *Specification of the Bluetooth System.*

4 Bluetooth v2.1 security

The level of a device in Bluetooth v2.1.is controlled by the device's security level. Depending on the security level set the processes and order of authorization, pairing and bonding are handled differently.

4.1 Bluetooth v2.1 security levels

In security mode 4 a device can assign one of the following security levels to an L2CAP Protocol Service Multiplexer (PSM) or an RFCOMM channel, as well as setting the default security level (applied to any service that has not had a security level specifically assigned). If not set by the user the default security level is set to Level 2. Level 0 security is only permitted for Service Discovery Protocol (SDP) services.

Table 4-1 Security level mapping to link key requirements

Security Level Requirement For Service	Link Key type required for remote devices that support Secure Simple Pairing (BTv2.1)	Link Key type required for remote devices that do not support Secure Simple Pairing (Pre Bluetooth v2.1)	Comments
Level 3 ■ Strong Man-in-the-middle protection required ■ Encryption required ■ User interaction acceptable	Authenticated	Combination	High Security
Level 2 ■ No Man-in-the-middle protection required ■ Encryption required	Unauthenticated	Combination	Medium Security

Table 4-1 Security level mapping to link key requirements (cont.)

Security Level Requirement For Service	Link Key type required for remote devices that support Secure Simple Pairing (BTv2.1)	Link Key type required for remote devices that do not support Secure Simple Pairing (Pre Bluetooth v2.1)	Comments
Level 1 ■ No Man-in-the-middle protection required ■ No encryption required ■ Minimal user interaction desired	Unauthenticated	None	Low Security (for example OPP)
Level 0 ■ No Man-in-the-middle protection required ■ No encryption required ■ No user interaction desired	None	None	Permitted only for SDP

4.2 Bluetooth v2.1 authorization messaging sequences

If the local device is initialized as v2.0, then authorization takes place after pairing:

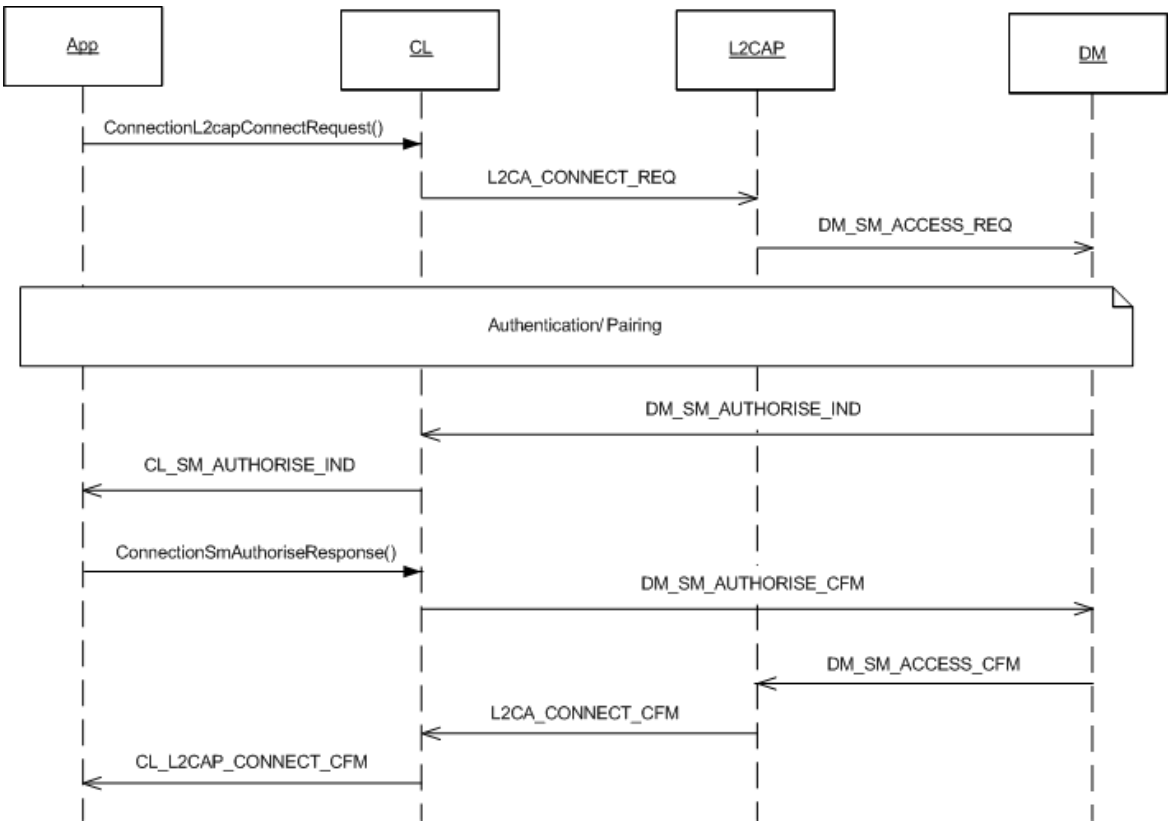


Figure 4-1 Authorization example, local legacy device (security mode 2)

If the local device is initialized as v2.1 the authorization takes place before pairing:

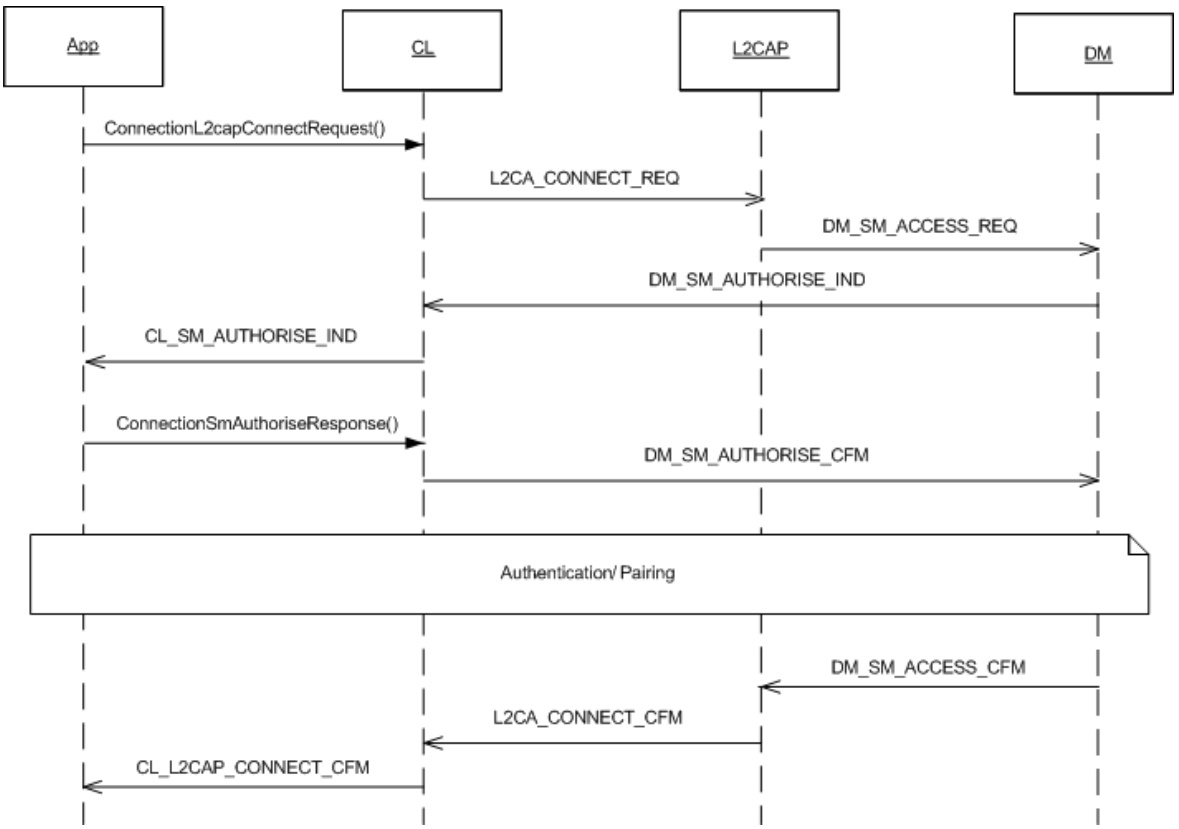


Figure 4-2 Authorization example, local v2.1 device (security mode 4)

An exception to this occurs if write authentication has been enabled on a v2.1 device. In which case, on determining that a remote device is a legacy device (using the `REMOTE_HOST_SUPPORTED_FEATURES_NOTIFICATION`) it pairs during link setup, that is, before service establishment and therefore before authorization. See Figure 4-4.

4.3 Bluetooth v2.1 dedicated bonding

A Bluetooth v2.1 device initiates bonding by calling the `ConnectionSmAuthenticate` function, see Figure 4-3.

When `ConnectionSmAuthenticate` is called the `connection` library deletes any existing link key associated with the remote device and initiates bonding. If the device has any services registered that

require Man-in-the-Middle (MITM) protection then the devices use one of the authentication models that require user interaction, see , otherwise the *just works* model (that is no user interaction) is used.

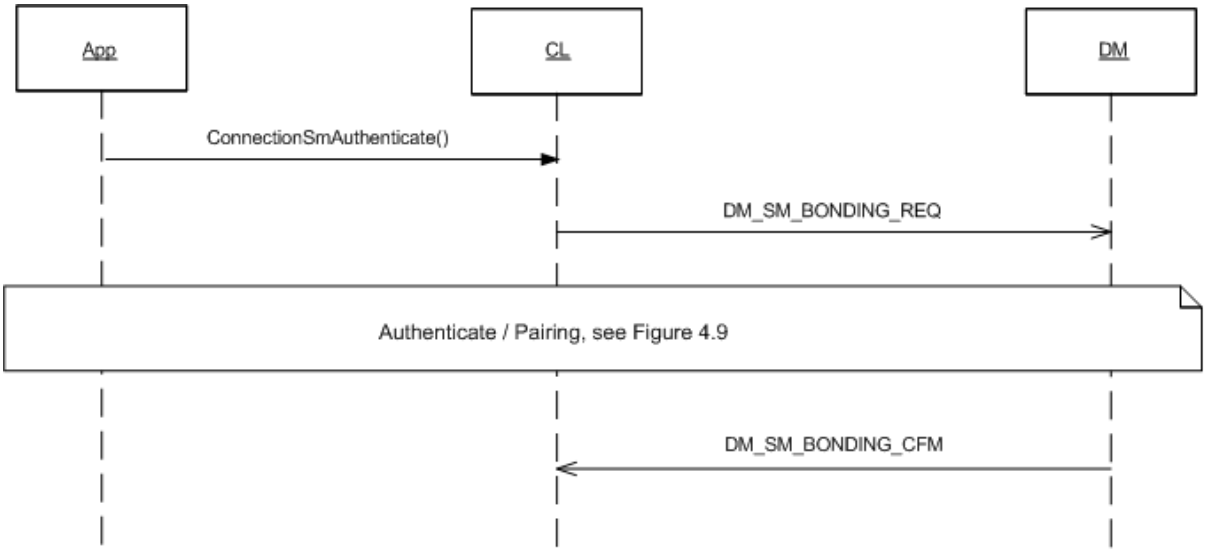


Figure 4-3 Dedicated bonding

NOTE A `CL_SM_AUTHENTICATE_CFM` message is sent to the application at the end of the Pairing message sequence, see Figure 4-9. The `DM_SM_BONDING_CFM` is consumed by the `connection` library as it contains no extra useful information.

The following message sequence chart assumes authorization has taken place.

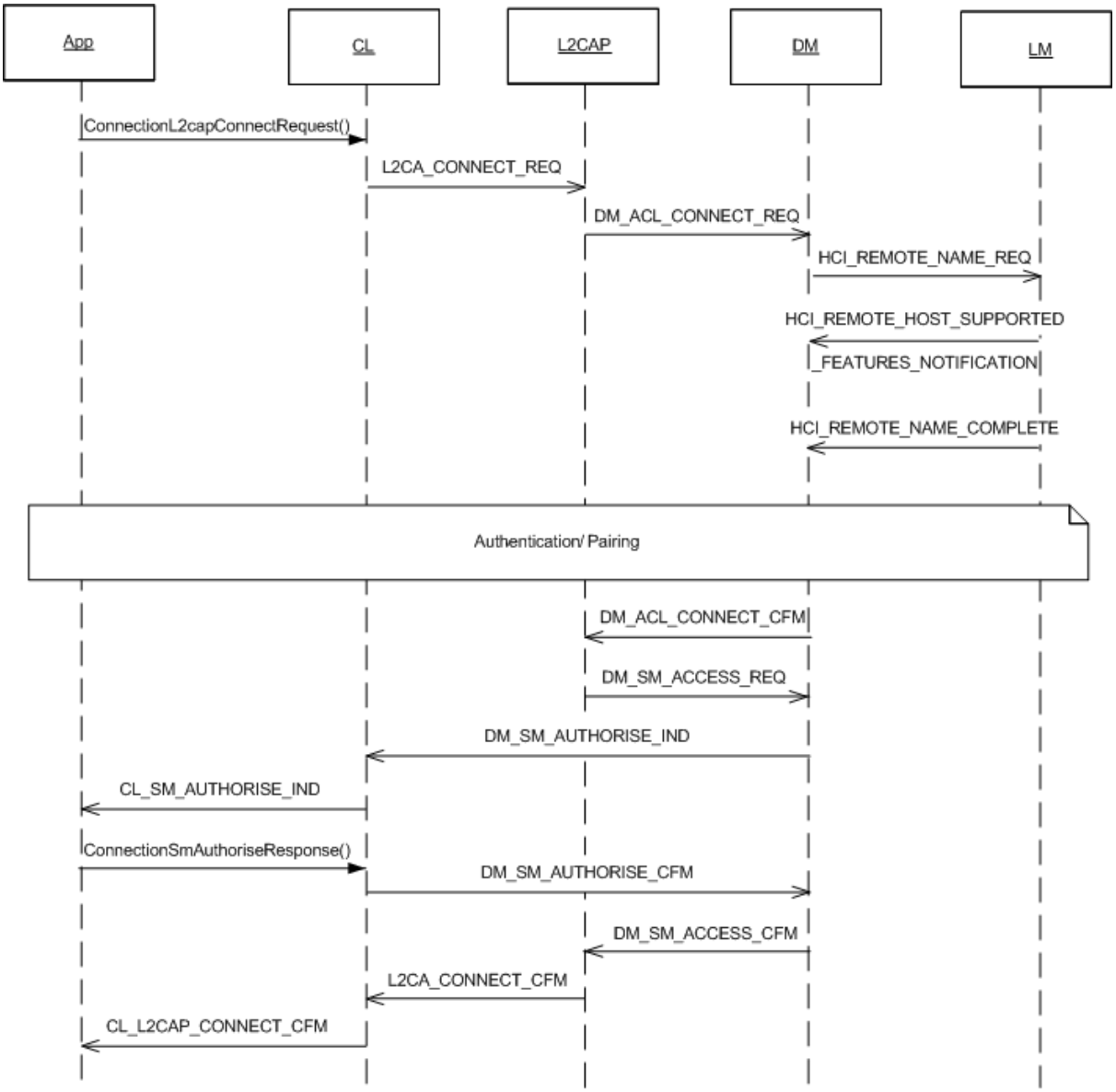


Figure 4-4 : Authorization example, local v2.1 device (security mode 4 with WAE) vs remote legacy device

NOTE If the remote device is v2.1 then write authentication is not used and authorization takes place as shown in Figure 4-2.

4.4 Bluetooth v2.1. authentication and pairing for L2CAP connections

Before connecting an L2CAP PSM that requires security (that is, not SDP), authentication/pairing takes place. Because the responding side does not know which PSM is being accessed until authentication/pairing has completed it may initiate a second pairing to upgrade the security level if the first pairing was not good enough for the service being accessed.

For example, if the initiator registers an L2CAP PSM as requiring security level 2 resulting in an unauthenticated pairing taking place, but the responding device has registered the PSM being connected to as requiring level 3 security, then on receiving the connection request another pairing is forced to achieve the higher security level.

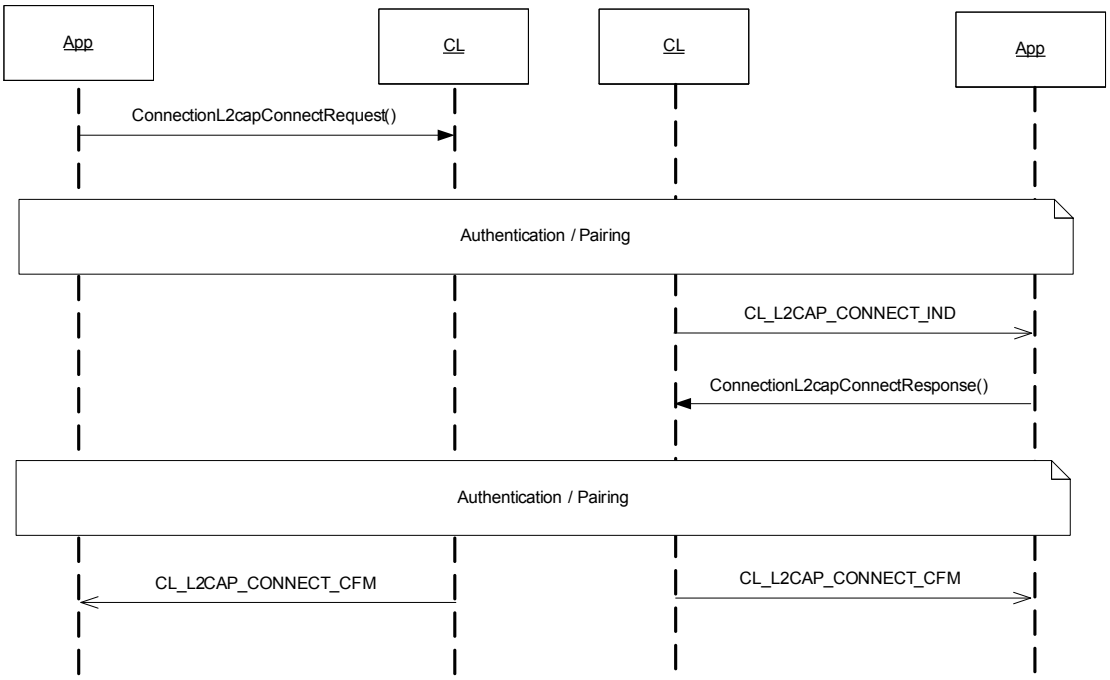


Figure 4-5 L2CAP authentication

4.5 Bluetooth v2.1. authentication and pairing for RFCOMM connections

Double pairing may occur when establishing an RFCOMM connection, where the RFCOMM multiplexer has a lower security level than the channel being accessed. This can be avoided on the responding side by requesting MITM protection if it is supported by the initiating device.

The initiating device can avoid double pairing either by performing a dedicated bonding before connecting, or by forcing MITM on the RFCOMM Multiplexer if it knows the channel it is connecting requires MITM. If the initiator can handle double pairing, then it should not force MITM unnecessarily.

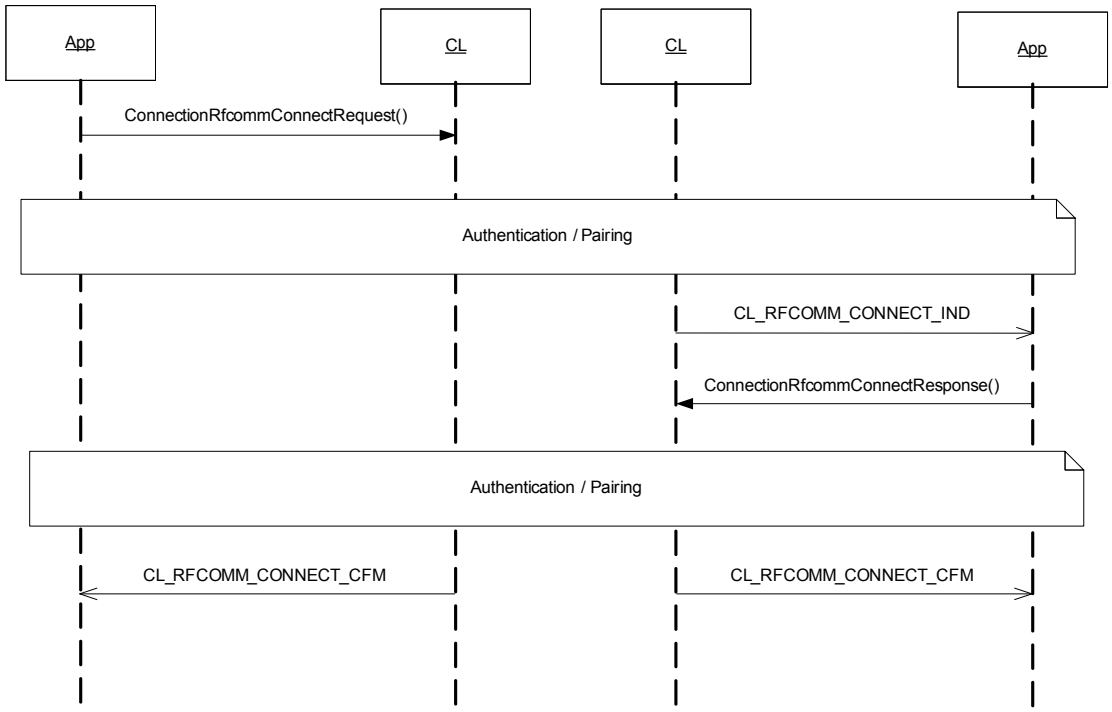


Figure 4-6 RFCOMM security

4.6 Authentication

On successfully pairing with a device, the associated link key is added to the Device Manager (DM) device database and stored in the Persistent Store.

NOTE The DM device database is lost when the chip is reset.

At initialization the `connection` library restores the database from entries in the Persistent Store. If a connection requires security the DM checks the device database for a link key associated with the remote device that has the security level required/expected or better. If a suitable link key is not found it sends a request for a key to the `connection` library. The `connection` library always responds

negatively to this request, as all devices in Persistent Store have been registered with the DM and no further checks are required.

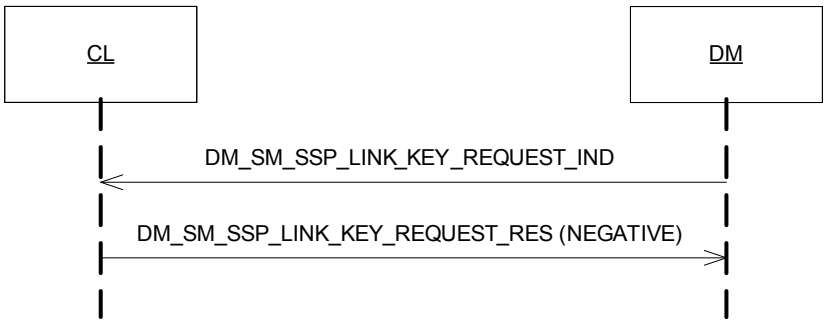


Figure 4-7 : Authentication

NOTE (1) When initiating a device for which a suitable link key is found then it is returned and the Headset attempts to authenticate the link.
(2) When initiating a device for which a suitable link key is not found then the Headset initiates pairing.
(3) If the Headset is the responding device and a suitable link key is not found then the remote device must be trying to authenticate.
(4) If the Headset is the responding device and a suitable valid link key is found, authentication succeeds.

4.7 Pairing

If the remote device is pre-2.1 then legacy pairing will be performed (that is, PIN entry), otherwise Secure Simple Pairing (SSP) is performed. The first stage of SSP is the IO capability exchange where the devices exchange their IO capability, MITM and bonding requirements and any OOB data that has previously been exchanged.

Both hosts receive a `CL_SM_IO_CAPABILITY_REQUEST_IND`, the initiating side is prompted first and should respond by calling `ConnectionSmIoCapabilityRequestRes()`. The remote device is then notified of the initiator's requirements via the `CL_SM_REMOTE_IO_CAPABILITY_IND` message and is then prompted to supply its requirements, which are sent back to the initiator.

Possible IO capabilities are:

- No input no output
- Display only
- Display yes/no
- Keyboard only
- Reject

These are used to determine what form of MITM interaction will be performed, if one is required. If MITM is required but unattainable, (for example, the devices are both display only devices), pairing fails.

A responding device in bondable mode should supply its IO capabilities and other requirements in its response.

A responding device in non-bondable mode should use a reject response at this stage if the initiator requests to bond. A responding device in non-bondable mode may send a reject response at this stage if the initiator does not request bonding, or it may supply its IO capabilities and other requirements as normal.

Force MITM: The application can specify a change to the MITM requirement at this stage, if a service is registered as not requiring MITM protection the application can state here that it wishes to force MITM protection anyway. However, it cannot force it not to use MITM protection if a service is registered as requiring it.

Bonding: The application can also specify its intent to bond with the remote device. This can be hard coded in the application, (for single profile devices or devices with no IO capability), or specified by the device user.

If the application specifies it intends to bond, then on completion of pairing the `connection` library registers the remote device and corresponding link key with the DM, and write it to the Persistent Store.

If the application specifies that it does not intend to bond, then on completion of pairing the `connection` library registers the remote device and corresponding link key with the DM and does not write it to the persistent store.

Out of Band Data: If the devices exchanged the optional hash C and rand R values via OOB inquiry (see 2), then the application should provide the remote OOB data at this point. To do this `oob_data_present` should be set to `TRUE` and `hash C` and `rand R` included in the response. If only hash C is present, then `rand R` should be zeroed out. If OOB pairing fails, then the data should be discarded and one of the other pairing message sequences used instead.

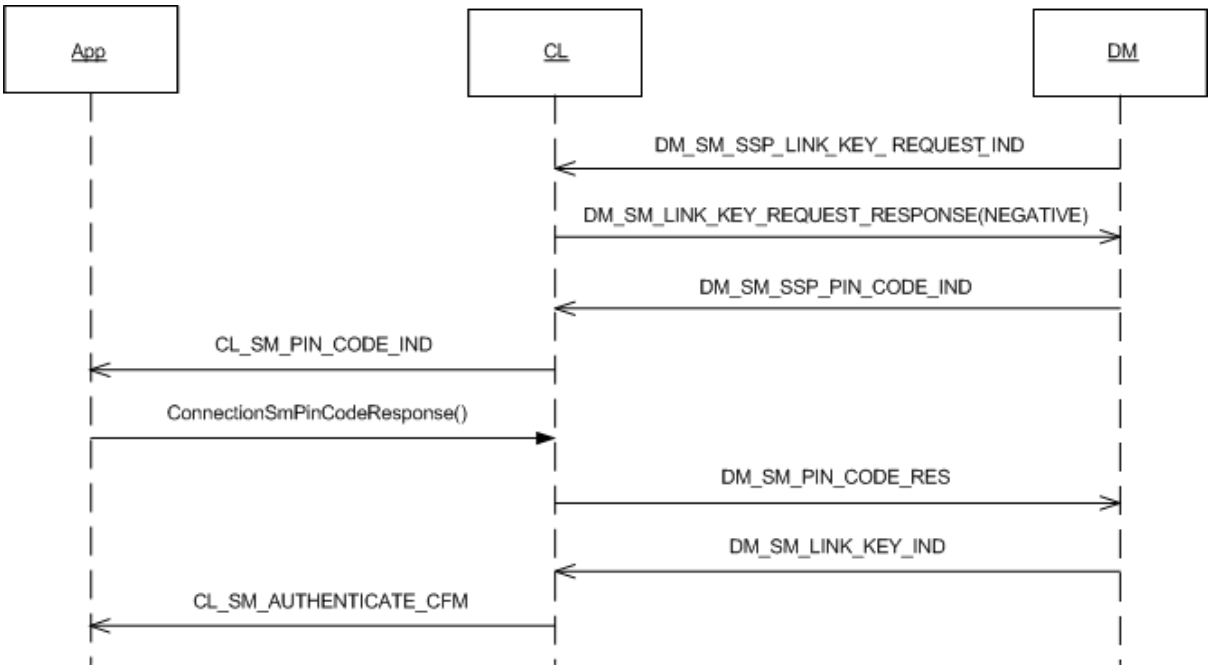


Figure 4-8 Legacy pairing

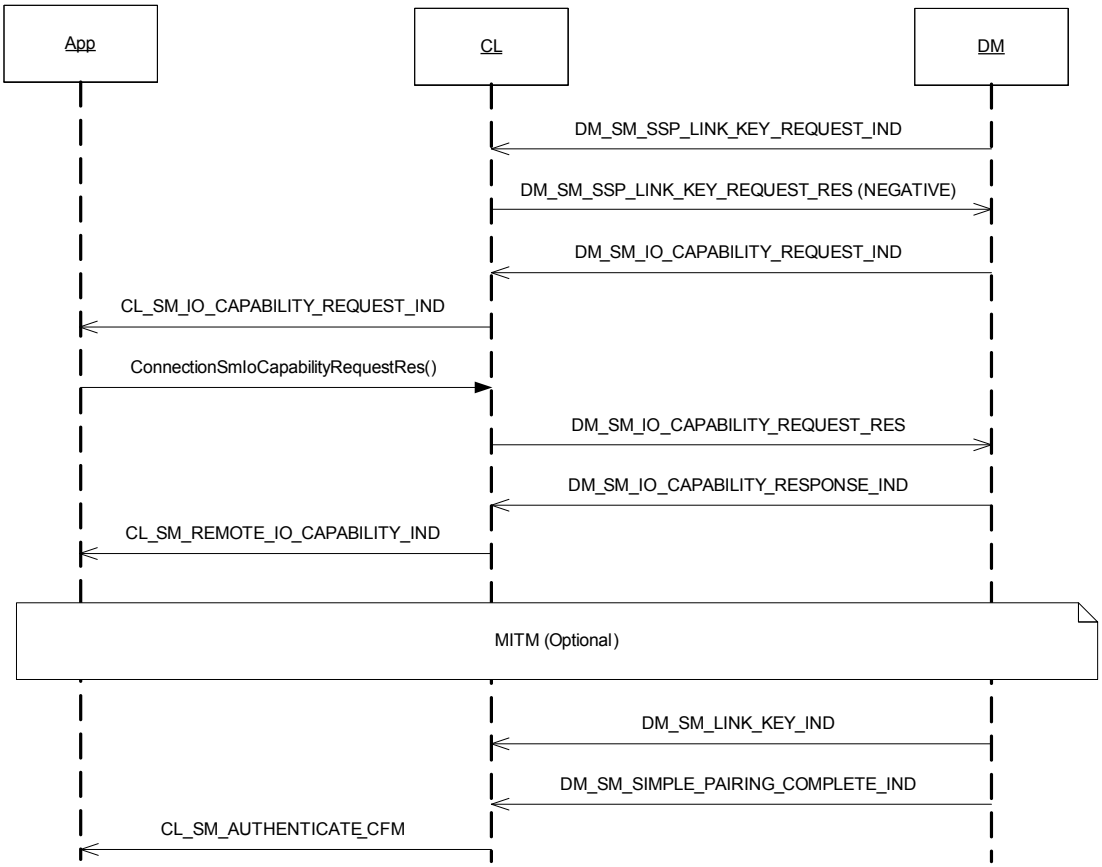


Figure 4-9 Secure simple pairing

4.7.1 MITM keyboard side passkey entry

Passkey entry occurs in two cases:

1. One device is a keyboard and one is a display
2. Both devices are keyboards

The keyboard device(s) receive a passkey request indication message and should then send keypress notifications to the remote device when the user enters data, (this is only useful if remote device is a display), followed by the passkey the user entered in the passkey request response.

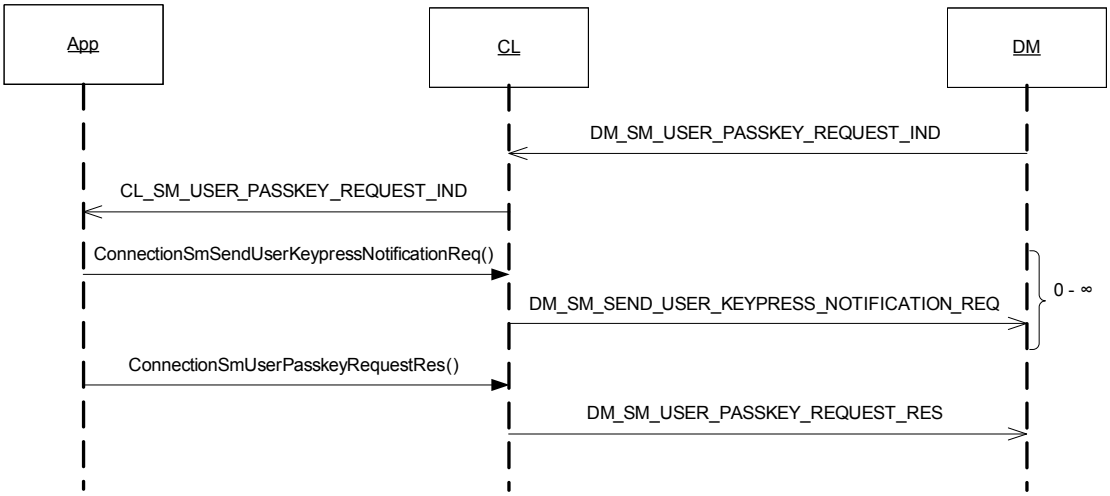


Figure 4-10 0 MITM keyboard side passkey entry

4.7.2 MITM display side passkey entry

On the display side the application receives a passkey notification indication message containing the passkey to be entered on the remote device. This value should be displayed until the `CL_SM_AUTHENTICATE_CFM` message is received. The display should also handle the different types of keypress notification that should follow the passkey.

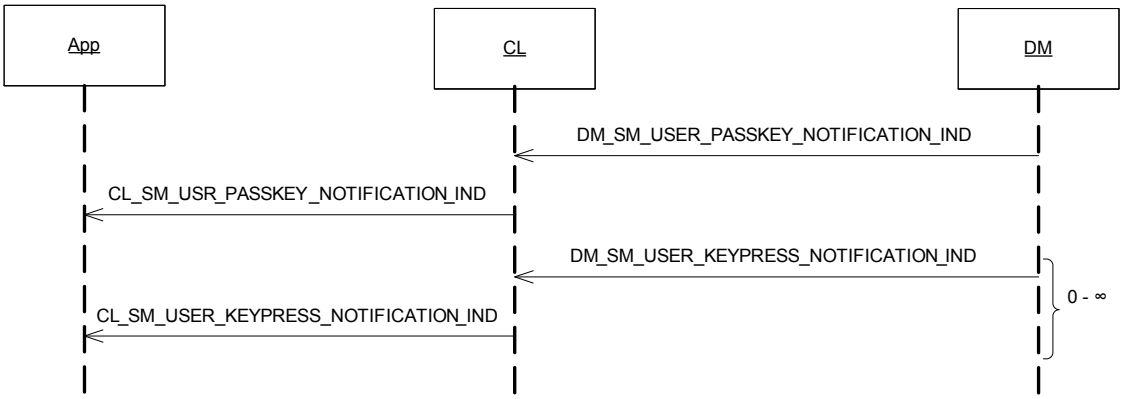


Figure 4-11 1 MITM display side passkey entry

4.7.3 MITM user confirmation

User confirmation occurs when both devices have a display *yes/no* capability. For both devices the application receives a confirmation request indication message containing a value to display to the user. The user should then confirm that the value displayed is the same on both devices.

A device that does not support MITM may in some cases receive a user confirmation request. In this case a reject response should always be sent.

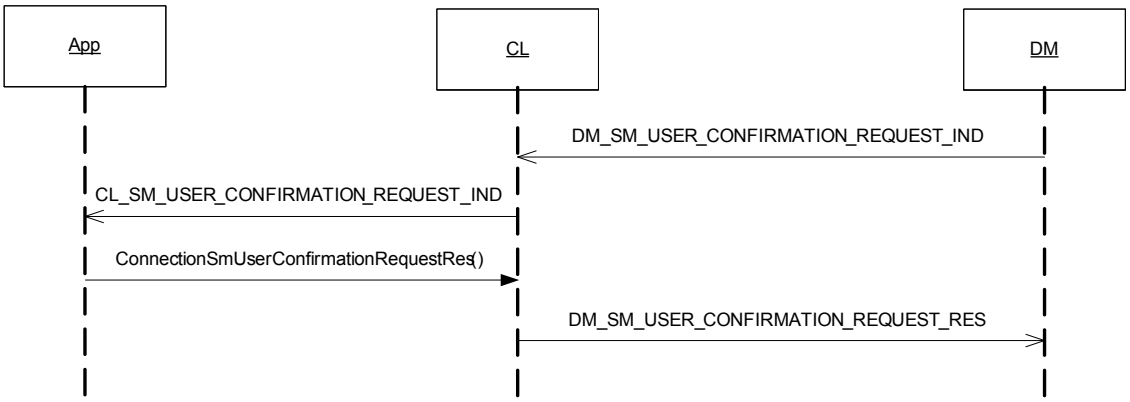


Figure 4-12 2: MITM user confirmation

4.8 Legacy issues

4.8.1 Write auth enable and SDP ping

Responding devices sometimes start a timer when an ACL has been established. If the timer fires before they are satisfied that something “worthwhile” has happened then the responder will close the ACL. Other timers may be started if SDP is open but unused, if the RFCOMM multiplexor is up but no channels are connected, and after the last L2CAP channel is disconnected. If these timers fire then they may cause the termination of the relevant L2CAP connection or of the ACL itself.

A “worthwhile” event could be the connection of an L2CAP channel, the configuration of an L2CAP channel, or perhaps even the first useful data transfer or AT command over a channel.

Unfortunately this means that some legacy devices will disconnect the ACL before pairing has completed, however we have two methods to prevent this:

- Enable write authentication. This means with legacy devices we will pair during link setup, so the pairing is finished before the ACL is established and thus the timer is never started. The circumstances under which write authentication is performed (that is, whether an existing ACL is dropped to enable pairing during link setup) can be set using `ConnectionSmSecModeConfig`.
- If write authentication cannot be used (either because the application has elected not to drop existing ACLs or write authentication is disabled) then to prevent the legacy remote device timing out we open an SDP channel and send repeated search requests (discarding the results). This ensures that the remote device is satisfied that something meaningful is happening and so stops or resets the timer.
  The second method (that is SDP ping) will not work if the local or remote device requires authentication on SDP connections. For this reason, it is strongly recommended that any v2.1 devices should not require security on outgoing SDP connections.
- In the current implementation of the firmware 23c upper layers an attempt to open an SDP channel to ping the remote device results in the access request being queued (despite it not requiring security). This means pinging the remote device is not possible until the pairing has completed, at which point the need to do so has passed with the result that timeouts with legacy devices may still occur. A fix for this is scheduled for 23d firmware.

4.8.2 Authentication failure, key missing

The way we handle a Key Missing failure, (that is, where we attempt to authenticate with an existing link key but the remote device has deleted its link key), has changed from v2.0. Previously a Key Missing failure would have resulted in one of the devices initiating pairing, however in v2.1 we treat this as a security failure and disconnect.

This is fine for outgoing connections, presumably if the user has deleted the 2.1 device from their phone they would not expect it to be able to reconnect without pairing with it again first.

Unfortunately, with incoming connections some legacy devices attempt to connect the service before authenticating the link, a practice not allowed in v2.1. Where the connecting device would have initiated pairing (as it had no link key) the responding v2.1 device attempts to authenticate the link before responding, but this fails with the Key Missing status which results in the connection being rejected.

For devices capable of notifying the user that they need to delete the link key and retry this is not a large problem, however some devices (for example a headset) do not have this capability. To avoid this problem pairing can be allowed after authentication failure with legacy devices.

For incoming connections, write authentication is not used and therefore does not prevent pairing after authentication failure.

4.9 Configuring security

Both write authentication and legacy pairing behavior can be set up using `ConnectionSmSecModeConfig`. This function can also be used to enable/disable the generation of debug link keys between two v2.1 devices, which allows recording of traffic over the air (for example by using a sniffer device) by using a known link key. This should never be enabled for purposes other than debug.

Document references

Document	Reference
Specification of the Bluetooth System	Core Specification v2.1 + EDR 26 July 2007

Terms and definitions

Term	Definition
ACL	Asynchronous ConnectionLess
BlueCore	Group term for the range of QTIL Bluetooth wireless technology ICs
Bluetooth	Set of technologies providing audio and data transfer over short-range radio connections
DM	Device Manager
EDR	Enhanced Data Rate
EIR	Extended Inquiry Response
IC	Integrated Circuit
IO	Input/Output
LM	Link Manager
MITM	Man-in-the-Middle
OOB	Out Of Band
OPP	Object Push Protocol profile
PSM	Protocol Service Multiplexer
QTIL	Qualcomm Technologies International, Ltd.
SDP	Service Discovery Protocol
SSP	Secure Simple Pairing
UUID	Universally Unique Identifier