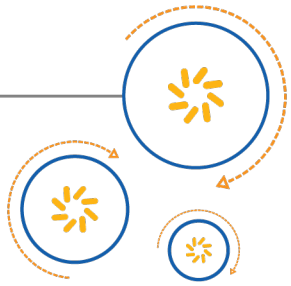
Qualcomm Technologies International, Ltd.



Connection Library RFCOMM API

Application Note

80-CT427-1 Rev. AH

October 22, 2017

Revision history

Revision	Date	Description
1	JUL 2010	Initial release. Alternative document number CS-00207134-AN.
2	AUG 2011	Updated to latest CSR™ style
3	MAY 2012	Updated to latest CSR style
4	JUL 2012	Reference to connection library header file update to `connection_no_ble.h` and minor formatting corrections
5	APR 2014	Updated to latest CSR style
6	FEB 2016	Updated to conform to QTI standards; No technical content was changed in this document revision
7	APR 2017	Updated for ADK 4.2. Added to the Content Management System.
AH	OCT 2017	Document Reference Number updated to use Agile number. No change to the technical content.

Contents

Figures

1 Creating an RFCOMM connection

See the ADK VM Library Documentation on the connection `connection_no_ble.h` header file for detailed information on the functions and structures referenced.

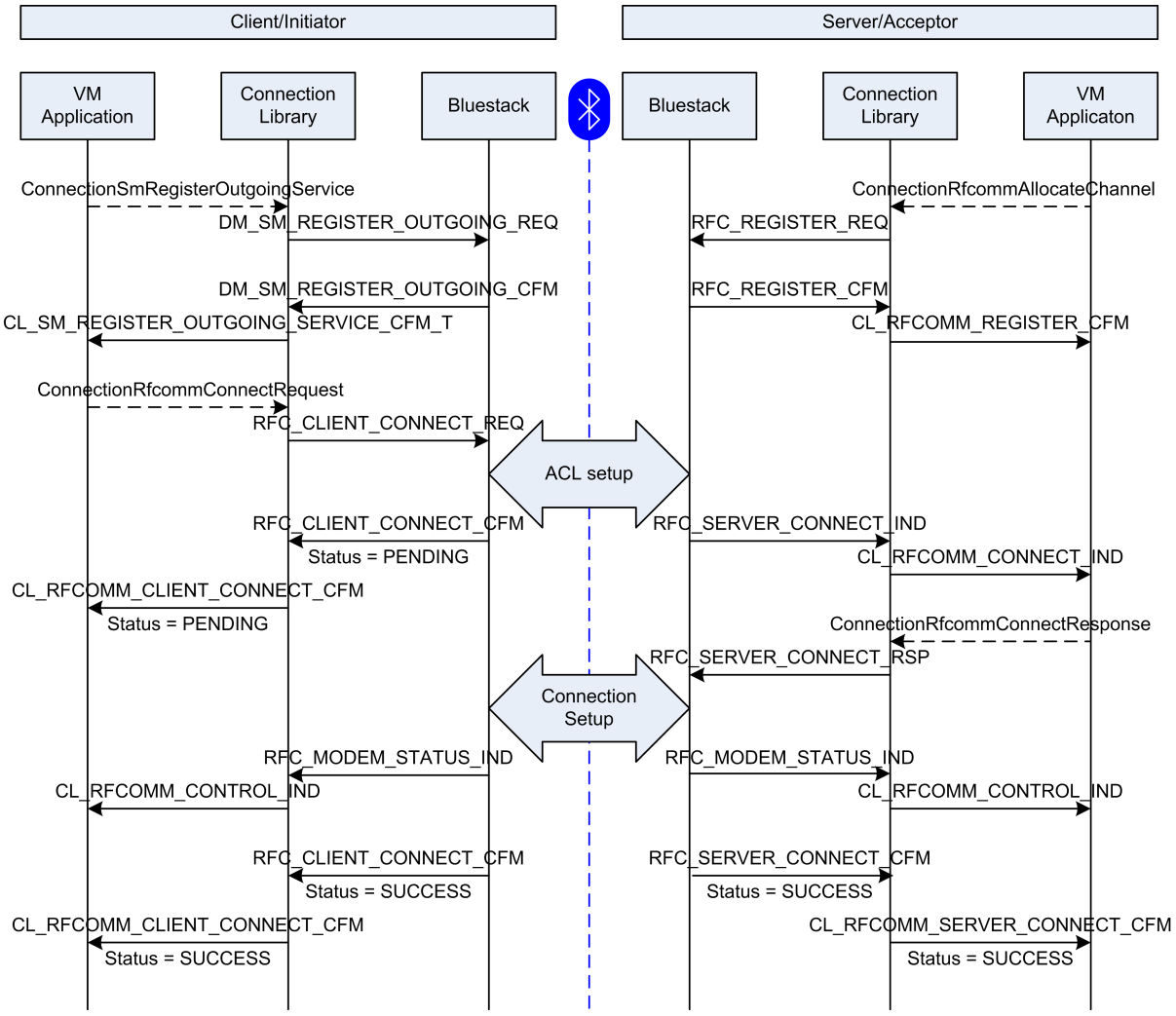


Figure 1-1 RFCOMM connection message sequence

NOTE In the MSC dashed arrows indicate function calls and solid arrows indicate messages.

1.1 Initiating RFCOMM connections

The Client/Initiator is the device that requests the RFCOMM connection.

Registering an outgoing service

For an outgoing RFCOMM connection, it is no longer necessary to allocate a server channel, although this can be done as before.

If a server channel is not used for the outgoing connection then the security of that connection must be specified using the `ConnectionSmRegisterOutgoingService()` function.

```
ConnectionSmRegisterOutgoingService(
      my_task,                          /* Task the CFM msg will go to. */
      remote_bdaddr,                    /* Remote device bd_addr. */
      protocol_rfcomm,                  /* dm_protocol_id */
      0,                                /* Suggested server channel. */
      sec4_out_level_1                  /* dm_security_out */
      );
```

Figure 1-2 Registering an outgoing service

The `ConnectionSmRegisterOutgoingService()` function, when set for the RFCOMM protocol and a suggested server channel of 0, results in a `CL_SM_REGISTER_OUTGOING_SERVICE_CFM` message being returned from BlueStack. The `CL_SM_REGISTER_OUTGOING_SERVICE_CFM` message contains a security channel ID set with the security defined in the `ConnectionSmRegisterOutgoingService()` function call. This security channel ID can then be used when making the RFCOMM Connection request, in the same way that a server channel ID would be.

Making an RFCOMM connection request

To initiate a connection, use the `ConnectionRfcommConnectRequest()` function.

```
ConnectionRfcommConnectRequest(
      my_task,                  /* Task the CFM msg will go to. */
      remote_bdaddr,            /* Remote device bd_addr. */
      security_channel,         /* Security / local server channel ID. */
      remote_server_channel,    /* Remote server channel to connect to. */
      0                         /* rfcomm_config_params. 0 = default */
      );
```

Figure 1-3 RFCOMM connect request

An rfcomm_config_params: Structure can be referenced for defining the following connection parameters:

- max_payload_size: Used per RFCOMM packet.
- modem_signal: Passed in a Modem Status message to the remote device during the connection.
- break_signal: Passed in a Modem Status message to remote device during the connection.
- msc_timeout: Time to wait to receive a Modem Status indication from the remote device before completing the connection anyway.

If NULL or 0 is passed instead of a pointer to an rfcomm_config_params structure, then default values are used (see the ADK VM Library Documentation on the connection_no_ble.h).

The sequence of messages received during a successful connection is:

- CL_RFCOMM_CLIENT_CONNECT_CFM, with an rfcomm_connect_status of rfcomm_connect_pending. This message also contains the Sink associated with the link, although it cannot be used to send or receive data until the RFCOMM connection is complete. The Sink can be used by the VM application to uniquely identify this particular RFCOMM connection.
- CL_RFCOMM_CONTROL_IND, containing modem_signal and break_signal parameters from the remote device. If BlueStack has not received a Modem Status message from the remote device before the msc_timout expires, it completes the connection without it and this message is not received.
- CL_RFCOMM_CLIENT_CONNECT_CFM, with an rfcomm_connect_status of rfcomm_connect_success. At this point the Sink returned in this message can be used to transfer data.

1.2 Accepting RFCOMM connections

The Server/Acceptor is the device that accepts an incoming RFCOMM connection

For an incoming RFCOMM connection, a local server channel is required for the remote device to connect with. Typically, the RFCOMM local server channel for an incoming connection is advertised in an SDP service record so that the remote device can find and connect to that service.

The ConnectionRfcommAllocateChannel() function can be used to allocate a local RFCOMM server channel.

```
ConnectionRfcommAllocateChannel(
      my_task,                  /* Task the CFM msg will go to. */
      0                         /* Suggested server channel. */
      );
```

Figure 1-4 Allocate RFCOMM server channel

If the suggested server channel is set to 0, then BlueStack returns the next available server channel starting from 1 and incrementing upwards. The assigned channel ID is returned in a CL_RFCOMM_REGISTER_CFM message.

Any other value, in the range 1 to 30, can be suggested if this has already been defined in a const SDP record for an RFCOMM service. If that local server channel ID is available, the same value is returned in the CL_RFCOMM_REGISTER_CFM message. If it is not available, then the server ID returned in the CL_RFCOMM_REGISTER_CFM message is the next available unused channel ID.

The security for this local RFCOMM server channel is the default set during Initialisation of the connection library that is, Mode 4, Secure Simple Pairing.

NOTE The security can be modified using the ConnectionSmSetSecurityMode() function (however, this is not covered in this document).

Accepting an incoming RFCOMM connection

A CL_RFCOMM_CONNECT_IND message initiates an incoming RFCOMM connection. This contains the Sink for the connection, which cannot be used for data transfer until the final CFM message indicating successful completion of the RFCOMM connection has been received.

The ConnectionRfcommConnectResponse() function must be used to accept an incoming connection indication.

```
ConnectionRfcommConnectResponse(
      my_task,                  /* Task this will be associated with. */
      (bool) TRUE,              /* Accept the connection */
      ind->sink,                /* Sink from CL_RFCOMM_CONNECT_IND */
      my_local_server_channel,  /* Local server channel ID (also in IND.*/
      0                         /* rfcomm_config_params. 0 = default */
     );
```

Figure 1-5 RFCOMM connect response

The rfcomm_config_parameters structure and values are the same as for the ConnectionRfcommConnectRequest() function.

The sequence of messages received during a successful connection is:

- CL_RFCOMM_CONTROL_IND, containing modem_signal and break_signal parameters from the remote device. Sending this during connection is optional. If BlueStack has not received a Modem Status message from the remote device before the msc_timout expires, it completes the connection without it and this message is not received.
- CL_RFCOMM_SERVER_CONNECT_CFM, with an rfcomm_connect_status of rfcomm_connect_success. At this point the Sink returned in the message can be used to transfer data.

2 Releasing an RFCOMM connection

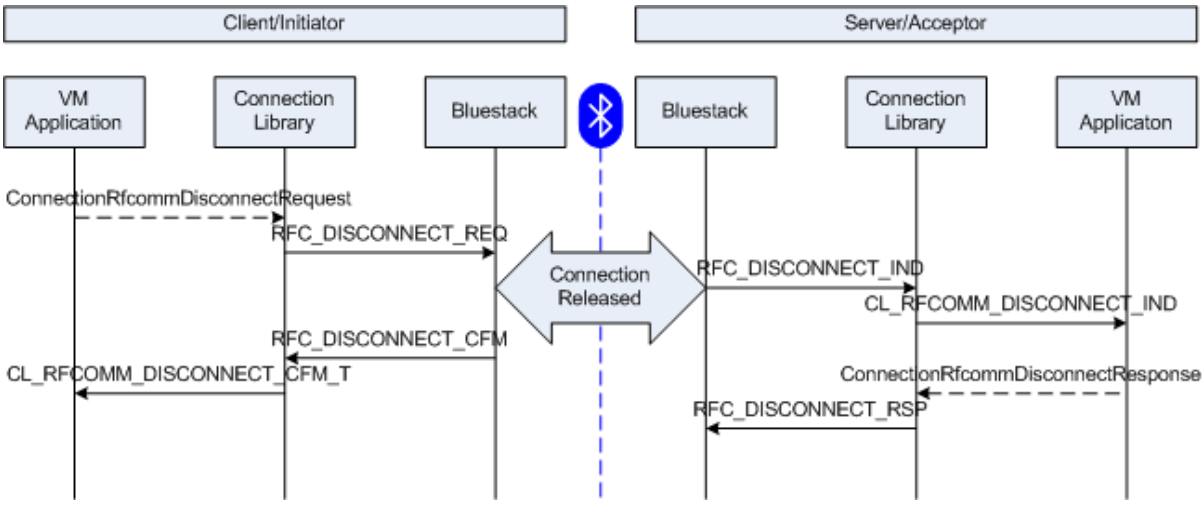


Figure 2-1 RFCOMM disconnection message sequence

Client/initiator

The Client/initiator is the device that initiates the release of an RFCOMM connection.

The function `ConnectionRfcommDisconnectRequest()` is used to disconnect an RFCOMM connection. The `Sink` that identifies the connection must be passed to this function.

```
ConnectionRfcommDisconnectRequest(
      my_task,            /* The task the CFM message will be sent to. */
      rfcomm_sink         /* The Sink identifying the RFCOMM link. */
      );
```

Figure 2-2 RFCOMM disconnect request

When this function has been called, the `Sink` identified is no longer valid and any data still in that `Sink`, or its related `Source`, is lost.

A `CL_RFCOMM_DISCONNECT_CFM` message is received in response with the status `rfcomm_disconnect_success`.

Server/acceptor

The Server/Acceptor is the device which receives the indication that an established RFCOMM connection has been disconnected by a remote device.

2.1 RFCOMM disconnection indication

A `CL_RFCOMM_DISCONNECT_IND` message indicates a disconnection by a remote device. It contains the Sink that identifies the RFCOMM connection that has disconnected.

The Sink and its associated Source remain until the `ConnectionRfcommDisconnectResponse()` function has been called to acknowledge the disconnection to the BlueStack layer. However, the link no longer exists so no data can be transmitted between devices although existing data in the Source buffer can still be read, until the `ConnectionRfcommDisconnectResponse()` function is called.

```
ConnectionRfcommDisconnectResponse(
      rfcomm_sink               /* The Sink identifying the RFCOMM link. */
      );
```

Figure 2-3 RFCOMM disconnect response

Terms and definitions

Term	Definition
API	Application Programming Interface
Bluetooth	Set of technologies providing audio and data transfer over short-range radio connections
CFM	Confirmation
IC	Integrated Circuit
ID	Identifier
IND	Indication
QTIL	Qualcomm Technologies International, Ltd.
RFCOMM	Radio Frequency COMmunications, serial port transport protocol over L2CAP
SDP	Service Discovery Protocol
VM	Virtual Machine