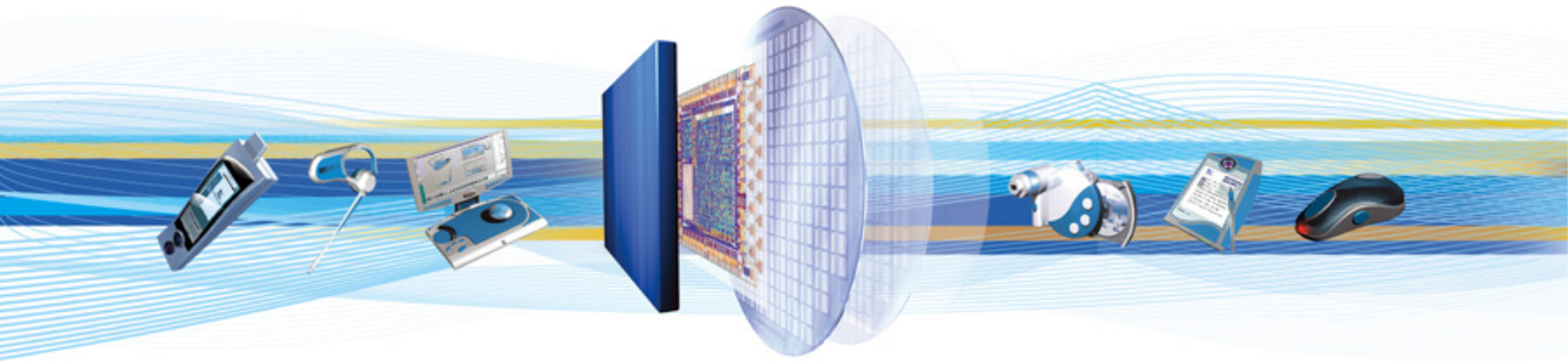
CSR Synergy Bluetooth 18.2.2

AMPM – Alternate MAC and PHY Manager

API Description

January 2012



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 CSR Synergy Bluetooth for using the Alternate MAC and PHY Manager (AMPM) – generally referred to as AMP Manager – as specified by the Bluetooth® Special Interest Group (SIG).

2 Description

2.1 Introduction

With the adoption of the Bluetooth Core Specification 3.0+HS (high speed) [1], the concept of using alternate MAC and PHYs (i.e. non-Bluetooth radios) for Bluetooth communication was introduced. The idea is to use a high throughput radio like IEEE 802.11 (Wi-Fi) for large bulk data transfers, which have a significantly higher bandwidth than the traditional EDR Bluetooth radio, which is limited at around 3mbps.

For example, for low-bandwidth links like browsing a remote file server a traditional Bluetooth radio link is more than sufficient. However, once the user starts transferring a large file e.g. a 100 MB movie trailer, the limited bandwidth on a Bluetooth radio starts to show. This is where AMP comes in; with AMP an existing Bluetooth connection can dynamically be *moved* from the Bluetooth radio to the AMP radio transparently for the end user. Once the transfer has completed, the connection can be moved back in order to save power.

2.2 Requirements

A BT3.0+HS system requires at least the following components to work:

- A Bluetooth radio with Secure Simple Pairing capability (i.e. at least a BT2.1 controller)
- An AMP radio – i.e. a 802.11 Wi-Fi radio, capable of Bluetooth AMP operation
- CSR Synergy Wi-Fi, configured with AMP support
- CSR Synergy Bluetooth, configured with AMP support

Apart from this, the peer system must also have a fully AMP-capable system with a compatible AMP radio in order for an AMP connection to be established between the CSR Synergy BT and peer systems.

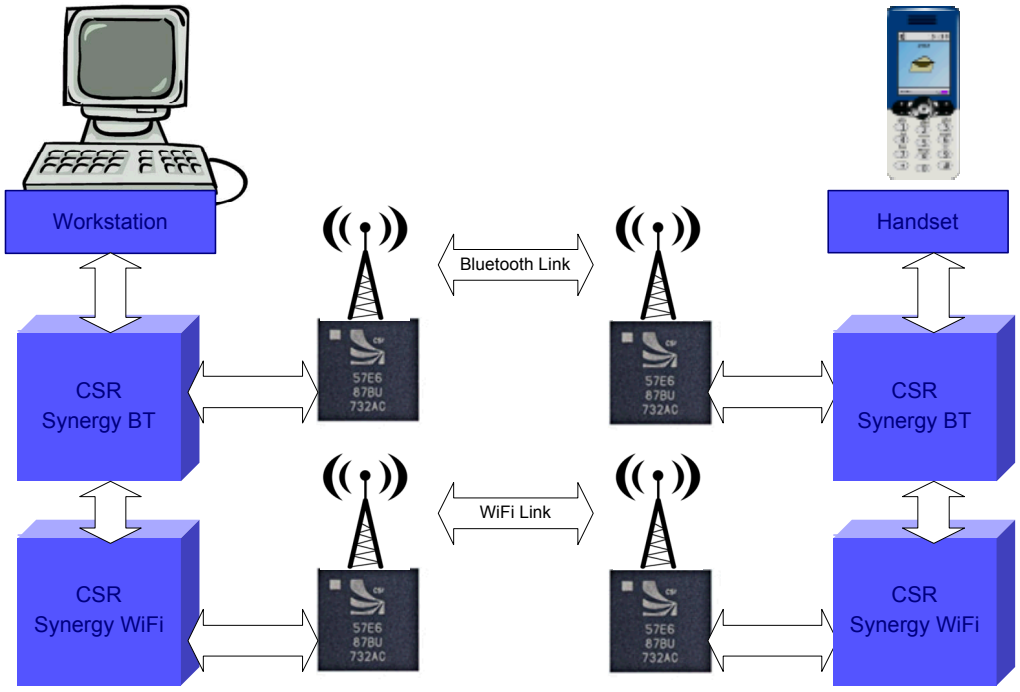


Figure 1: Typical AMP setup example

The AMPM module, which is an integrated part of CSR Synergy Bluetooth, contains all the necessary software components to be able to control AMP capable radios via a CSR proprietary software HCI emulation layer. The

CSR Synergy Wi-Fi component likewise supports this interface, making it possible to seamlessly integrate CSR Synergy Bluetooth and CSR Synergy Wi-Fi into a Bluetooth® 3.0+HS AMP capable system.

Note that in Figure 1, the type of device (i.e. computer and handset) is not important in regards to whether the device(s) can run AMP. As long as a device has at least a BR/EDR BT2.1 controller and a set of compatible AMP controllers, it may be possible to establish a high-speed AMP connection between the two devices.

2.3 Reference Model

The reference model for an AMP system is depicted in Figure 2 below.

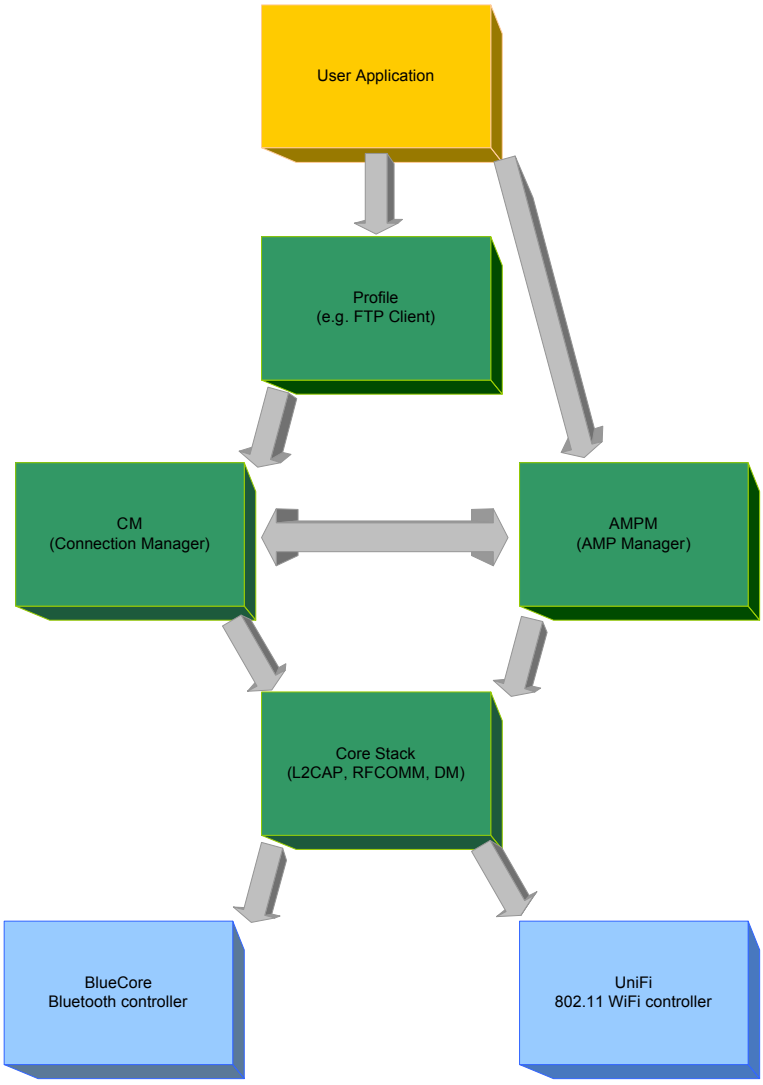


Figure 2: AMP Manager System reference model

This API document describes how the User Application interfaces to the AMP Manager. The other interfaces are handled internally in CSR Synergy Bluetooth. Blue boxes are hardware, green boxes are components provided by CSR Synergy Bluetooth, and yellow boxes are to be implemented by the application developer.

2.4 Typical Use Case

The typical use case for AMP is to facilitate an existing Bluetooth Profile with high speed data transfers, for example the OBEX File Transfer Protocol. No matter what profile is used with AMP, the AMP connection procedure is the same:

1. The user performs a normal Bluetooth connection procedure, i.e.
    a. Inquiry and remote peer selection
    b. Service search
    c. Pairing (secure simple pairing)
2. The user connects the desired service, for example FTP
    a. While this is happening, the AMPM will automatically interrogate the peer device to discover a remote AMP manager. If found, it will also discover remote AMP radios
3. The application is notified of possible compatible AMP radios
    a. The application stores the possible AMP radios – or it may request a list at any later point
4. The user browses the remote FTP server's directories and files, and initiates a file transfer of a large file
5. The application requests the AMPM to move the FTP connection from BR/EDR to AMP.
    a. The AMPM performs the connection move procedure while the file transfer is still in progress. The only user-visible change is the speed-up once the AMP connection is used.
6. The file transfer completes, and the application moves the FTP connection back to BR/EDR.

With a properly designed application, the use of AMP can be fully automatic: If the application identifies that a sufficiently large file is being transferred, the move from BR/EDR to AMP can happen automatically – as can the automatic move back to BR/EDR once the use of AMP is complete.

The CSR Synergy Bluetooth AMPM has been designed to make it as easy as possible for the application to control the use of AMP: The API consists of four types of procedures:

- Registration and un-registration.
  Before the application can utilise AMP, it must register each connection with the AMPM. An AMP registration is based on the global CSR Synergy Bluetooth connection identifier, known as a CsrBtConnId
- AMP controller discovery
  The AMPM can be configured to automatically notify the application of possible AMP radios, or the application can actively request a remote AMP discovery
- AMP move channel
  Once a set of compatible local and remote AMP radios has been found, the application can request the AMPM to move a connection from BR/EDR to AMP, or vice versa. Any type of connection or profile can be used over AMP, as long as it has been registered and thus have a valid global connection identifier (CsrBtConnId)
- AMP controller power on
  If a local AMP controller is available for Bluetooth operation but physically power down, the AMPM can request a registered application to power-on the AMP controller when required for moving a connection.

Below, typical scenarios of all four procedures are shown.

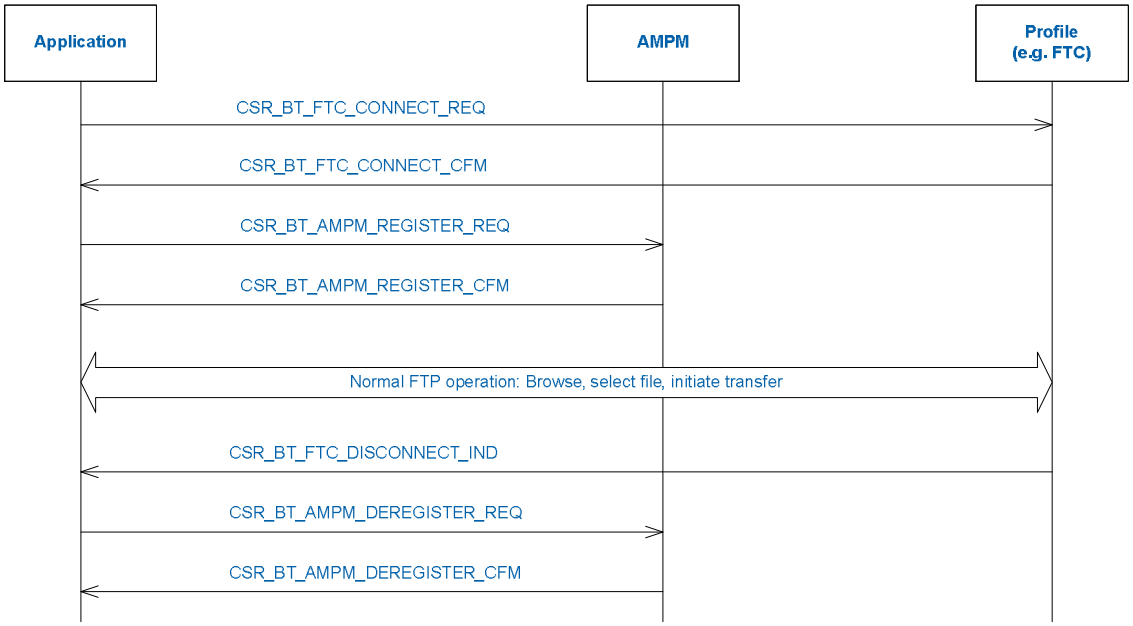


Figure 3: Example of how the application registers and unregisters with the AMPM

In Figure 3, a typical AMPM registration and un-registration procedure is shown: First, the application brings up a normal Bluetooth profile connection. Once the normal connection is established, the application registers the global Bluetooth connection identifier (the CsrBtConnId) with the AMPM. The CsrBtConnId is contained in all Bluetooth CONNECT_CFM and/or CONNECT_IND primitives.

When the profile connection is shut down (in the above, the peer disconnects), the application must unregister the AMPM registration again.

In Figure 4 below shows how the AMP controller listing and move channel initiator procedure works. Figure 5 shows how the move procedure works for the responder. Both figures assume that a profile connection is already established, and that global a connection identifier of that connection has been registered with the AMPM.

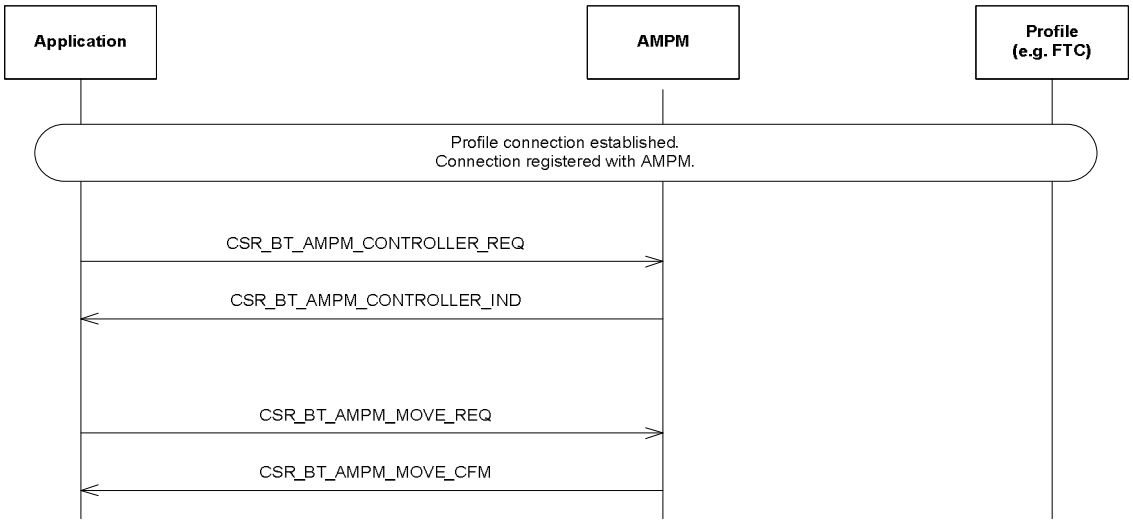


Figure 4: Example of AMP controller discovery and channel move initiation

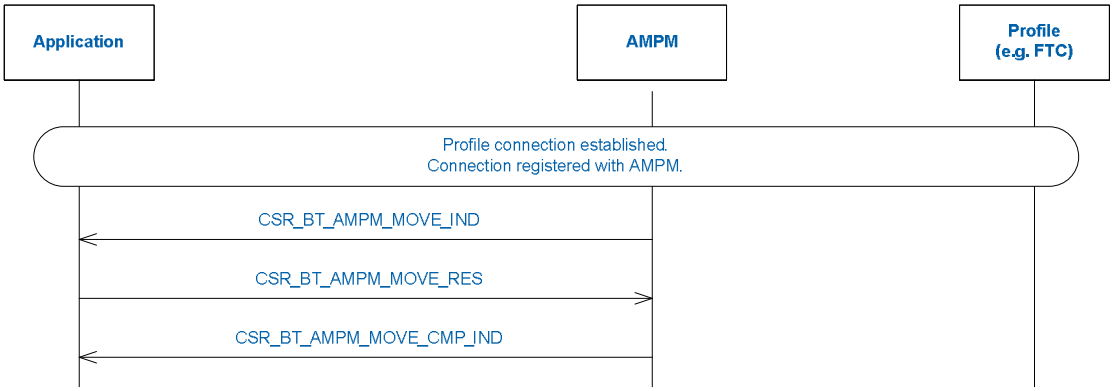


Figure 5: Example of AMP move channel as responder

The AMP controller discovery uses two primitives: `CSR_BT_AMPM_CONTROLLER_REQ` and `CSR_BT_AMPM_CONTROLLER_IND`. The indication primitive may contain the discovery AMP radios (known as AMP controllers) that can be used for carrying the profile connections.

The application can opt to automatically receive AMP controller list updates when it registers with the AMPM. This means that the application does not actively have to send the `CSR_BT_AMPM_CONTROLLER_REQ`. Instead, the AMPM will monitor all Bluetooth connections, and after the automatic retrieval of peer AMP information the AMPM will send the `CSR_BT_AMPM_CONTROLLER_IND` to the application, if any compatible AMPs were found.

The channel move procedure is fairly straight forward; if the application is the initiator a normal request/confirm primitive pair is used. If the application is the responder, i.e. the peer initiated the move procedure the application will receive a complete indication – the `CSR_BT_AMPM_MOVE_CMP_IND` – once the move procedure is complete. The AMPM protocol will handle move procedure collisions internally.

Note that an AMP move channel procedure only ever relates to a *single* profile connection. Moving one connection will not affect other connections to the same or other devices.

A BR/EDR Bluetooth connection will always be present in an AMP setup – even if all profile connections have been moved to AMP. This is needed as the BT3.0+HS specification requires all L2CAP protocol communication to only ever use the BR/EDR controller. CSR Synergy will automatically attempt to put the BR/EDR link in to a power-conserving and coex-friendly sniff mode when an AMP channel is in use.

When a local AMP controller is available for Bluetooth operation but physically powered down prior to moving a connection, an application must register to receive and handle power-on indications from the AMPM using the `CSR_BT_AMPM_REGISTER_POWER_ON_REQ` primitive. When an AMP controller power on is required, the AMPM will send a `CSR_BT_AMPM_POWER_ON_IND` to the registered application which should respond with a `CSR_BT_AMPM_POWER_ON_RES` when the power on is completed or rejected/failed.

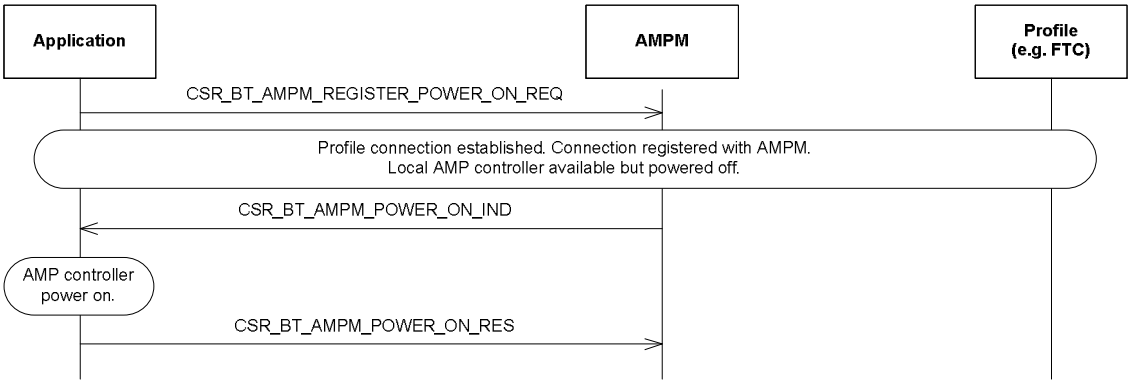


Figure 6: Example of AMP controller power on

Only one application may register to handle the AMP power on indications. If no application is registered when a move happens for a locally powered off AMP controller, the move will fail.

2.5 AMP Best Practices

One of the most common concerns about AMP is *when* it makes sense to move a channel from BR/EDR to AMP and vice versa. Unfortunately there is no simple answer to this, as it depends on the particular use case that the application is trying to solve – and what overall requirements an AMP device must fulfil. For example

- **Low power**
  Generally speaking, an AMP controller uses less power-per-bit than the BR/EDR controller once a connection has been established and when the utilisation of the AMP radio link is high. However, for idle links or low-throughput links where the BR/EDR sniff mode can be used, the BR/EDR controller is usually far superior power-wise to any AMP controller.
- **High throughput**
  AMP controllers generally have much higher throughput than BR/EDR, so when optimising for throughput, the application should favour the AMP controller over BR/EDR.
- **Sustained traffic or data bursts**
  For example, file transfer profiles and applications generally have a clear notion of when high bandwidth is required; either the profile is transferring data, or it is not transferring data. Other profiles like PAN or DUN doesn't have the same notion of data bursts; they maintain a channel for an extended period of time and there is generally no way for the profile to know what sort of data or bandwidth requirements make sense at any given point in time. Only data throughput measurements of opaque data may give an indication of whether BR/EDR or AMP would be most suitable.
- **Latency**
  The end-to-end latency is generally lower on AMP links, so for critical low-latency applications it may make sense to prefer AMP over BR/EDR.
- **Warm-up time**
  If an AMP controller is completely powered off (for example in order to save power), it may take several seconds for the radio to power up and initialise all the hard- and software components needed to drive the AMP controller. During this time, it may be possible for the BR/EDR controller to complete the required operation, rendering the AMP controller useless once it is ready.
- **AMP is used by other profiles**
  If an AMP link is already in use by one profile, it probably makes sense to move all other profiles to the AMP controller.
- **AMP controller operating in native mode**
  Some AMP controllers – 802.11 in particular – does not only support Bluetooth AMP but also have a native mode of operation, for example TCP/IP traffic over 802.11. When such controllers are in native operation mode, it may not be possible to also run AMP on the controller.

When deciding on whether to use BR/EDR or AMP, the application developer should try to keep all these points in mind and balance or prioritise them against each other.

The AMPM does not yet monitor all profiles for all of these parameters, nor does it include *intelligence* to autonomously move channels between BR/EDR and AMP. However, the AMPM API has been designed to allow this sort of intelligence to be added at a later revision. For example, the AMPM already contains functionality to automatically move channels as soon as an AMP link is available (see section 3.1 and section 3.8).

Until this intelligence has been implemented, the application needs to decide whether or not to use AMP. CSR recommends a few basic rules for *when* to use AMP:

- If the total transfer size of e.g. a file transfer exceeds 1MB, use the AMP controller
- If the total amount of objects to be transferred is known and is larger than 100, use the AMP controller
- If the total, sustained bandwidth requirement exceeds 1.5mbps, use the AMP controller
- If an AMP link is already in use, move any non-idle links to the AMP controller as well
- If none of the above matches, use the BR/EDR controller

2.6 AMP Controllers

In order to use AMP, both the local device and the remote peer device must have two compatible AMP controllers. When a local and a remote AMP controller match, this is known as a *local – remote AMP controller pair;* this pair consists of two numbers of the type `CsrBtAmpController`, which is what it reported to the application during the AMP controller discovery. The local-remote controller pair is also what is used in the AMP move channel procedure to designate what radio target that should be used.

AMP controller IDs are unique for each device, so no assumptions must be made on what numbering that is used. I.e. the application **must not** guess hard-code or in any other way invent AMP controller IDs. Only controller IDs reported from the AMPM can be used. Also note that there is no relationship between local and remote AMP controller IDs.

There are two special values for the AMP controller ID pair; `CSR_BT_AMP_CONTROLLER_BREDR` (=0), which always designate the BR/EDR controller. This value can always be used, and is always both present and valid. The second special value is `CSR_BT_AMP_CONTROLLER_UNKNOWN` (=0xFFFF). This value is used for indicating that no AMP controller ID could be obtained, i.e. it is the invalid value.

2.7 Profiles and Updated Profiles Optimised for AMP

In order for a profile to fully utilise AMP, the profile specification generally has to be updated such that the specification

- Uses L2CAP enhanced retransmission mode (ERTM) or streaming mode (SM)
- Uses L2CAP as the transport protocol and multiplexer
- Allows the L2CAP FCS (frame check sequence) to be disabled when AMP is used

These profile requirements does not explicitly rule out AMP utilisation for any existing profile, but it does impose problems with the RFCOMM stream protocol used by all legacy OBEX profiles and other serial port emulation profiles such as DUN. RFCOMM has its own flow control and multiplexer system, i.e. all RFCOMM profiles run on top of a single L2CAP connection. As AMP channel usage is controller on the L2CAP level, this forces RFCOMM to either run *all* or *none* of the RFCOMM channels over AMP. This legacy system design *may* introduce serious IOP problems, so CSR does *not* recommend the use of RFCOMM over AMP.

Instead of RFCOMM, the OBEX-over-L2CAP enhancements for GOEP should be used. This enhancement allows each OBEX channel to run directly on top of L2CAP and no longer depend on RFCOMM. Not only is this much more efficient, but it also allows each OBEX profile to be controlled individually in regards to AMP.

Whenever an AMP connection is in use, CSR Synergy Bluetooth will automatically attempt to put the BR/EDR connection in to a power-conserving and coex-friendly sniff mode.

CSR Synergy Bluetooth does allow RFCOMM channels to be used with AMP, however, as RFCOMM has it's own multiplexer system and uses only one L2CAP channel for all RFCOMM channels, there are a number of limitations to RFCOMM-over-AMP: Only one RFCOMM channel may request or respond to channel move procedures towards a designated peer at a time. The AMPM will internally ensure this consistency.

3 Interface Description

This chapter documents the public API of the AMPM by means of message sequence charts (MSCs), function prototype descriptions and signal members.

3.1 Registration

Before a profile connection can be used with AMP, the connection must exist and then be registered with the AMPM. This means that the application first establishes a normal profile connection which will provide the application with the global Bluetooth connection identifier of the particular connection. The identifier is always of type `CsrBtConnId`, and the parameter name is usually called `btConnId`.

To register a connection with the AMPM, the application uses the `CSR_BT_AMPM_REGISTER_REQ` primitive. Once the AMPM has processed the request, a `CSR_BT_AMPM_REGISTER_CFM` is sent back to the application. A MSC of the signalling is depicted in Figure 7.

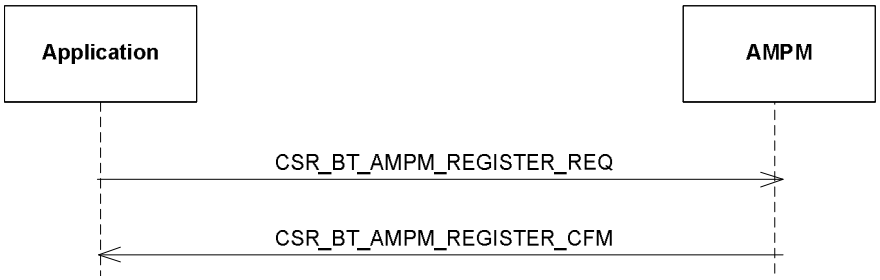


Figure 7: AMPM registration

In order to send the `CSR_BT_AMPM_REGISTER_REQ` primitive, the `CsrBtAmpmRegisterReqSend()` function is used. The function takes the arguments described in Table 2.

Type	Argument	Description
CsrSchedQid	qid	Handle to application, i.e. the queue on which the CSR_BT_AMPM_REGISTER_CFM and subsequent upstream primitives are sent

CsrUint32	flags	Controls special AMPM behaviour. The follow bitmask values can be OR'ed together:
		
		`CSR_BT_AMPM_FLAGS_NONE`
		(disable all flags)
		
		`CSR_BT_AMPM_FLAGS_AUTO_MOVE_ACCEPT`
		(automatically accept peer move requests)
		
		`CSR_BT_AMPM_FLAGS_AUTO_MOVE_ALWAYS`
		(automatically initiate a move procedure on AMP availability)

Type	Argument	Description
CsrUint32	eventMask	What events AMPM to subscribe for. The following bitmask values can be OR'ed together: CSR_BT_AMPM_EVENTS_DISABLE_ALL (disable all events) CSR_BT_AMPM_EVENTS_ENABLE_ALL (enable all events, including future features) CSR_BT_AMPM_EVENTS_AUTO_DISCOVERY (perform automatic discovery and inform application of results) CSR_BT_AMPM_EVENTS_AUTO_MOVE (subscribe to automatic move procedure results)
CsrBtConnId	btConnId	Global Bluetooth connection identifier. This value is usually supplied via a profile CONNECT_CFM or CONNECT_IND.
CsrBtDeviceAddr	addr	Address of the remote peer device.

Table 1: Arguments for `CsrBtAmpmRegisterReqSend` function

When the AMPM has processed the registration request, a CSR_BT_AMPM_REGISTER_CFM primitive will be sent back to the application. The primitive members are described in Table 2.

Type	Member	Description
CsrBtAmpmPrim	type	Signal identity – always set to CSR_BT_AMPM_REGISTER_CFM
CsrUint32	eventMask	Event mask that was registered in the REQ.
CsrBtConnId	btConnId	Global Bluetooth connection identifier this register confirm is for – i.e. the value passed in the REQ.
CsrBtDeviceAddr	addr	Address of the remote peer device.
CsrBtResultCode	resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier = CSR_BT_SUPPLIER_AMPM then the possible result codes can be found in csr_bt_ampm_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.
CsrBtSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 2: Members in a CSR_BT_AMPM_REGISTER_CFM primitive

Note: Before the controller discovery and/or move channel procedures can be used, a connection *must* be registered with the AMPM.

Note: It is possible to register with the AMPM with `btConnId = CSR_BT_CONN_ID_INVALID` (=0). Doing so will allow the application to request controller listings, but as no valid connection is ever associated with the invalid global connection identifier, it makes no sense to attempt to move this identifier.

Note: If any of the CSR_BT_AMPM_FLAGS_AUTO_MOVE flags are enabled, the application will only be notified of automatic move channel procedures if the CSR_BT_AMPM_EVENTS_AUTO_MOVE event mask flag is set. Automatic move results are conveyed to the application by means of a CSR_BT_AMPM_MOVE_AUTO_CMP_IND, see section 3.8.

3.2 Un-Registration

When a profile connection that has been registered with the AMPM is disconnected, the connection must also be un-registered with the AMPM in order for the AMPM to release any resources allocated for the connection.

To un-register a connection with the AMPM, the application uses the CSR_BT_AMPM_DEREGISTER_REQ primitive. Once the AMPM has processed the request, a CSR_BT_AMPM_DEREGISTER_CFM is sent back to the application. A MSC of the signalling is depicted in Figure 8.

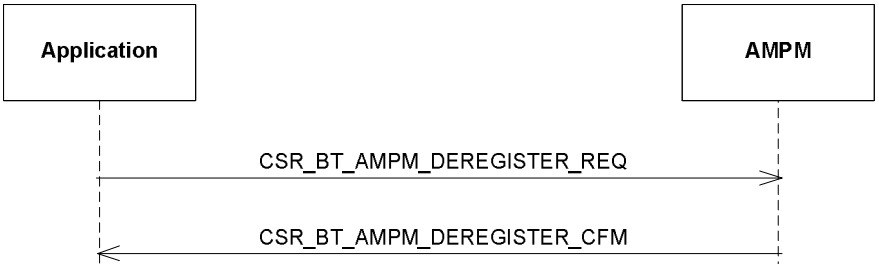


Figure 8: AMPM un-registration

In order to send the CSR_BT_AMPM_DEREGISTER_REQ primitive, the CsrBtAmpmDeregisterReqSend() function is used. The function takes the arguments described in Table 3.

Type	Argument	Description
CsrSchedQid	qid	Handle to application, i.e. the value previously passed in the CSR_BT_AMPM_REGISTER_REQ.
CsrBtConnId	btConnId	Global Bluetooth connection identifier previously passed in the CSR_BT_AMPM_REGISTER_REQ.
CsrBtDeviceAddr	addr	Address of the remote peer device previously passed in the CSR_BT_AMPM_REGISTER_REQ.

Table 3: Arguments for CsrBtAmpmDeregisterReqSend function

When the AMPM has processed the un-registration request, a CSR_BT_AMPM_DEREGISTER_CFM primitive will be sent back to the application. The primitive members are described in Table 4.

Type	Member	Description
CsrBtAmpmPrim	Type	Signal identity – always set to CSR_BT_AMPM_DEREGISTER_CFM
CsrBtConnId	btConnId	Global Bluetooth connection identifier that has been unregistered.
CsrBtDeviceAddr	addr	Address of the remote peer device that has been unregistered.
CsrBtResultCode	resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier = CSR_BT_SUPPLIER_AMPM then the possible result codes can be found in csr_bt_ampm_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.
CsrBtSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 4: Members in a CSR_BT_AMPM_DEREGISTER_CFM primitive

3.3 Registration of Power-On Handler

If a local AMP controller can be available for Bluetooth operation but physically powered down it is required to register an application handler responsible for performing AMP controller power-on. Only one application handler may be registered.

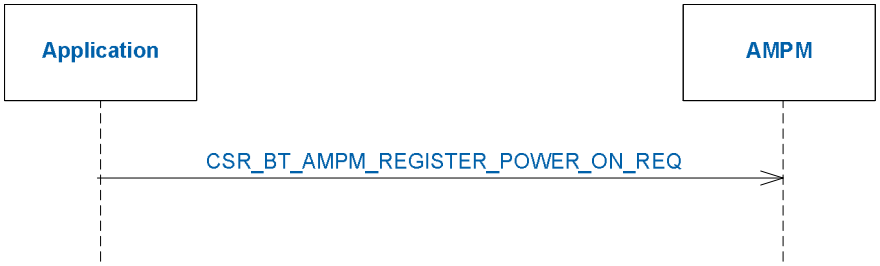


Figure 9: AMP controller power-on handler registration

To send the CSR_BT_AMPM_REGISTER_POWER_ON_REQ primitive, the CsrBtAmpmRegisterPowerOnReqSend() function is used. The function takes the arguments described in Table 5.

Type	Argument	Description
CsrSchedQid	qid	Handle to application responsible for AMP controller power-on.

Table 5: Arguments for CsrBtAmpmRegisterPowerOnReqSend function

There is no confirmation primitive returned.

3.4 AMP Controller Power-On

When the AMPM needs to power-on a local AMP controller, it sends a CSR_BT_AMPM_POWER_ON_IND to the application handler previously registered. The application may choose to reject the request to perform the power-on. The application must respond to the indication within 15 seconds when the power-on is initiated by a peer device. The application must complete the power-on before returning the response.

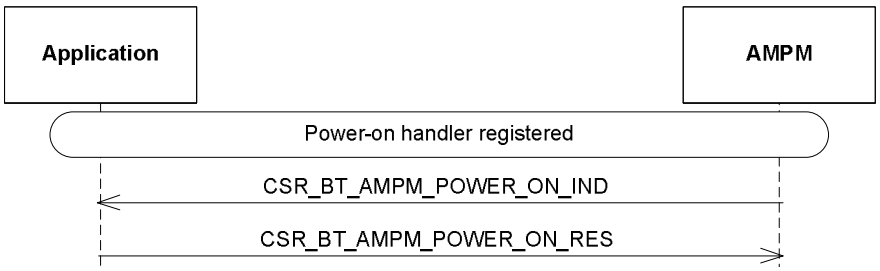


Figure 10: AMP controller power-on

The CSR_BT_AMPM_POWER_ON_IND primitive members are described in Table 6.

Type	Member	Description
CsrBtAmpmPrim	Type	Signal identity: CSR_BT_AMPM_POWER_ON_IND
CsrBtAmpAmpControllerType	ampType	Hardware type. See the csr_bt_amp_hci.h file #defines for the various AMP hardware types.
CsrBtAmpController	localId	Local AMP controller ID.
CsrBool	localInit	Set to TRUE when the cause of power-on was the local device initiating a move. Set to FALSE when initiated by peer.

Table 6: Members in a CSR_BT_AMPM_POWER_ON_IND primitive

When the power-on hander receives the CSR_BT_AMPM_POWER_ON_IND primitive it must decide if it the AMP controller should be powered on or not, this might involve user interaction. If powered on, it must be completed and responded to by sending a CSR_BT_AMPM_POWER_ON_RES primitive within 15 seconds from the arrival of the indication, otherwise the move will fail for peer initiated move attempts.

To send a CSR_BT_AMPM_POWER_ON_RES primitive, the CsrBtAmpmPowerOnResSend() function is used. The function takes the arguments described in Table 7.

Type	Argument	Description
CsrBtAmpAmpControllerType	ampType	Hardware type as set by the indication.
CsrBtAmpController	localId	Local AMP controller ID as set by the indication.
CsrBool	complete	Set to TRUE if the AMP controller power-on was successfully completed, FALSE otherwise.

Table 7: Arguments for CsrBtAmpmPowerOnResSend function

3.5 Controller Discovery

Controller discovery is the procedure that interrogates the remote peer for compatible AMP radios, and reports the possible AMP radios to the application as a local and remote AMP controller ID pair. A pair of controller IDs can then subsequently be used as a target for an AMP move channel procedure.

The AMP controller discovery can either be automatic or manual. The automatic discovery is enabled if the AMPM registration was done with the CSR_BT_AMPM_EVENTS_AUTO_DISCOVERY event mask flag set. In this case, the AMPM will automatically send up CSR_BT_AMPM_CONTROLLER_IND if and only if valid AMP controller pairs are found. The automatic signaling is depicted in Figure 11.

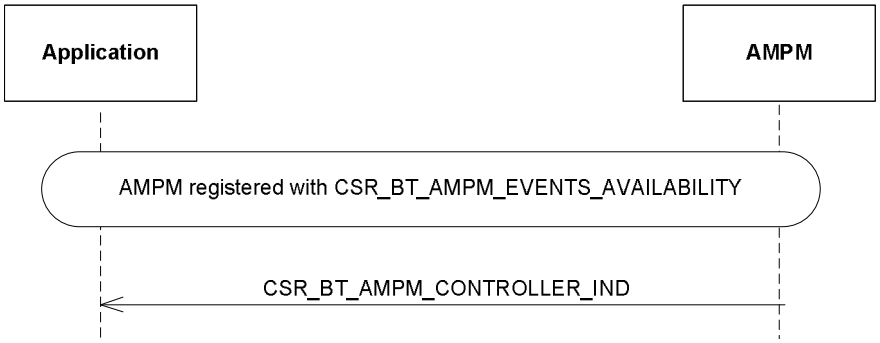


Figure 11: AMPM automatic discovery

To use manual controller discovery the application sends a CSR_BT_AMPM_CONTROLLER_REQ to the AMPM. This method can be used even if automatic discovery is enabled. A manual discovery will generate a CSR_BT_AMPM_CONTROLLER_CFM as a response. The manual discovery signaling is depicted in Figure 12.

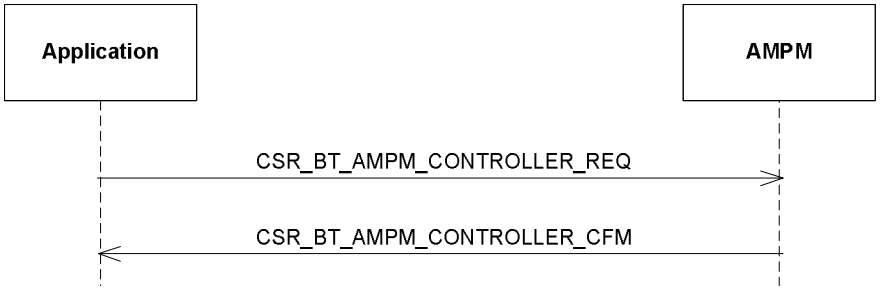


Figure 12: AMPM manual discovery

In order to send the manual discovery request, i.e. the CSR_BT_AMPM_CONTROLLER_REQ primitive, the CsrBtAmpmControllerReqSend() function is used. The function takes the arguments described in Table 8.

Type	Argument	Description
CsrSchedQid	qid	Handle to application, i.e. the value previously passed in the CSR_BT_AMPM_REGISTER_REQ.

Type	Argument	Description
CsrBtDeviceAddr	addr	Address of the remote peer device previously passed in the CSR_BT_AMPM_REGISTER_REQ.
CsrBtConnId	btConnId	Global Bluetooth connection identifier previously passed in the CSR_BT_AMPM_REGISTER_REQ.

Table 8: Arguments for `CsrBtAmpmControllerReqSend` function

On automatic discovery, the AMPM will send a CSR_BT_AMPM_CONTROLLER_IND to the application, and for manual discovery it wil send a CSR_BT_AMPM_CONTROLLER_CFM. Both primitives share the same structure with only the *type* member being different. The common structure is explained in Table 9.

Type	Member	Description
CsrBtAmpmPrim	Type	Signal identity: Automatic discovery: CSR_BT_AMPM_CONTROLLER_IND Manual discovery: CSR_BT_AMPM_CONTROLLER_CFM
CsrBtConnId	btConnId	Global Bluetooth connection identifier previously passed in the CSR_BT_AMPM_REGISTER_REQ.
CsrBtDeviceAddr	addr	Address of the remote peer device that has been unregistered.
CsrUint8	ampsCount	Number of elements in the *amps* array
CsrBtAmpmControllerEntry*	amps	Pointer to array of AMPM controller pairs. See Table 10 for a description of the members.
CsrBtResultCode	resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier = CSR_BT_SUPPLIER_AMPM then the possible result codes can be found in csr_bt_ampm_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.
CsrBtSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 9: Members in the CSR_BT_AMPM_CONTROLLER_IND and CSR_BT_AMPM_CONTROLLER_CFM primitives

The CSR_BT_AMPM_CONTROLLER_IND/CFM *amps* arrays have the following structure:

Type	Member	Description
CsrBtAmpAmpControllerType	ampType	Hardware type. See the csr_bt_amp_hci.h file #defines for the various AMP hardware types.
CsrBtAmpAmpStatus	status	Remote AMP status and Bluetooth availability. See the csr_bt_amp_hci.h file #defines for the status values.
CsrUint32	caps	Reserved. Always 0.
CsrBtAmpController	localId	Local AMP controller ID.
CsrBtAmpController	remoteId	Remote AMP controller ID.

Table 10: Members in a CsrBtAmpmControllerEntry result structure

Note: The *amps* pointer must be freed by the application when it is no longer needed.

Note: The *resultSupplier* and *resultCode* fields are only used for the manual AMP discovery. When automatic discovery is used, the controller indication has already successfully completed so in this case, the supplier will be CSR_BT_SUPPLIER_AMPM and the result code will be CSR_BT_AMPM_RESULT_SUCCESS.

3.6 Move Channel as Initiator

Once a local-remote AMP controller ID pair has been obtained, the application can attempt to move the profile connection to/from any valid BR/EDR or AMP radio. The AMPM will automatically perform all necessary steps to complete the move; both the physical and logical link channel establishment including the L2CAP move protocol will all be done internally. The application will only be informed of the final result once the move has either successfully completed, or failed. If the move failed, normal operation may continue on the old radio.

To initiate a move channel procedure, the application uses the `CSR_BT_AMPM_MOVE_REQ` primitive. Once the AMPM has processed the request and performed all necessary steps to complete the move procedure, a `CSR_BT_AMPM_MOVE_CFM` is sent back to the application. A MSC of the signalling is depicted in Figure 13.

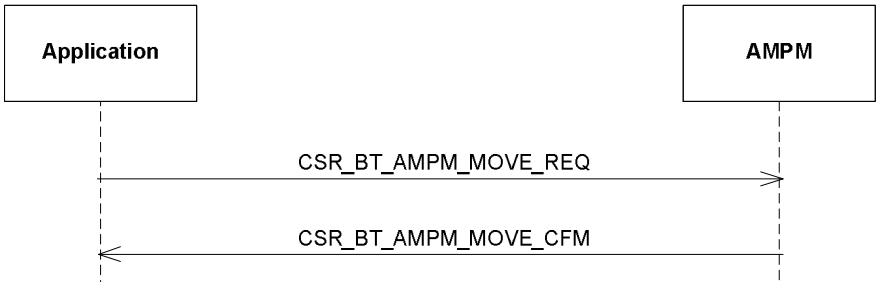


Figure 13: AMPM move channel as initiator

In order to send the `CSR_BT_AMPM_MOVE_REQ` primitive, the `CsrBtAmpmMoveReqSend()` function is used. The function takes the arguments described in Table 11.

Type	Argument	Description
CsrBtConnId	btConnId	Global Bluetooth connection identifier previously passed in the `CSR_BT_AMPM_REGISTER_REQ`.
CsrBtAmpController	localId	Local AMP controller ID – obtained from the `CSR_BT_AMPM_CONTROLLER_IND`.
CsrBtAmpController	remoteId	Remote AMP controller ID – obtained from the `CSR_BT_AMPM_CONTROLLER_IND`.

Table 11: Arguments for `CsrBtAmpmMoveReqSend` function

When the AMPM has completed the move channel procedure, a `CSR_BT_AMPM_MOVE_CFM` primitive will be sent back to the application. The primitive members are described in Table 12.

Type	Member	Description
CsrBtAmpmPrim	Type	Signal identity – always set to CSR_BT_AMPM_MOVE_CFM
CsrBtConnId	btConnId	Global Bluetooth connection identifier that has been moved.
CsrBtAmpController	localId	Local AMP controller ID that is now in use for this channel.
CsrBtResultCode	resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier = CSR_BT_SUPPLIER_AMPM then the possible result codes can be found in csr_bt_ampm_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.
CsrBtSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 12: Members in a CSR_BT_AMPM_MOVE_CFM primitive

Note: If the application registered with the `CSR_BT_AMPM_FLAGS_AUTO_MOVE_ALWAYS` flag set, the application does not need to send the CSR_BT_AMPM_MOVE_REQ, as the AMPM will automatically initiate a

move procedure on behalf of the application. The application can, however, still send the request and it will be handled automatically.

3.7 Move Channel as Responder

If an AMP capable peer device requests a move channel procedure, the local device has to decide whether to allow the channel to be moved. Once the move channel procedure has completed the local device will be informed about the final result code.

When the peer requests a move channel, the AMPM will send a `CSR_BT_AMPM_MOVE_IND` primitive to the application. The application *must* then accept or reject the move using a `CSR_BT_AMPM_MOVE_RES` primitive. Finally, when the move channel procedure has completed, the AMPM will pass up the final result codes to the application in a `CSR_BT_AMPM_MOVE_CMP_IND` primitive. A MSC of the signalling is depicted in Figure 14.

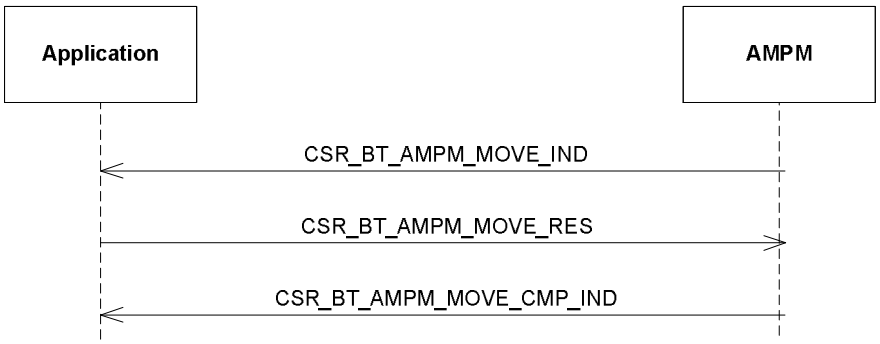


Figure 14: AMPM move channel as responder

The move channel indication (that originates from the peer system), contains the primitive members described in Table 13.

Type	Member	Description
CsrBtAmpmPrim	Type	Signal identity – always set to CSR_BT_AMPM_MOVE_CFM
CsrBtConnId	btConnId	Global Bluetooth connection identifier of the channel, which the peer wants to move.
CsrBtAmpController	localId	Local AMP controller ID, which the peer wants to move the channel to.

Table 13: Members in a CSR_BT_AMPM_MOVE_IND primitive

Once the move channel indication has been received by the application, the application must reply and either accept or reject the move by means of a `CSR_BT_AMPM_MOVE_RES` primitive. In order to send this primitive, the `CsrBtAmpmMoveResSend()` function is used. The function takes the arguments described in Table 14.

Type	Argument	Description
CsrBtConnId	btConnId	Global Bluetooth connection identifier previously passed in the `CSR_BT_AMPM_REGISTER_REQ`.
CsrBool	accept	TRUE if application accepts the move, FALSE if it rejects.

Table 14: Arguments for `CsrBtAmpmMoveResSend` function

When the AMPM has completed the move channel responder procedure, a `CSR_BT_AMPM_MOVE_CMP_IND` primitive will be sent back to the application. The primitive members are described in Table 15.

Type	Member	Description
CsrBtAmpmPrim	Type	Signal identity – always set to CSR_BT_AMPM_MOVE_CMP_IND
CsrBtConnId	btConnId	Global Bluetooth connection identifier that has been moved.

Type	Member	Description
CsrBtAmpController	localId	Local AMP controller ID that is now in use for this channel.
CsrBtResultCode	resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier = CSR_BT_SUPPLIER_AMPM then the possible result codes can be found in csr_bt_ampm_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.
CsrBtSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 15: Members in a CSR_BT_AMPM_MOVE_CMP_IND primitive

Note: The CSR_BT_AMPM_MOVE_CMP_IND and CSR_BT_AMPM_MOVE_CFM primitive types are the same – only the signal identifiers (the *type* member) are different

Note: If the peer attempts to move a channel of which the global connection identifier (the CsrBtConnId) has not been registered with the AMPM, the AMPM will automatically reject the move without informing the application (as there is no application registered).

Note: If the application registered with the CSR_BT_AMPM_FLAGS_AUTO_MOVE_ACCEPT flag set, the application will *never* receive a CSR_BT_AMPM_MOVE_IND as the AMPM will automatically accept the peer move request on behalf of the application.

3.8 Automatic Move Channel Results

If any of the CSR_BT_AMPM_FLAGS_AUTO_MOVE flags were set during registration of an AMP channel, the AMPM has been allowed to either automatically accept peer move requests – or to automatically initiate a move channel procedure as soon as a set of compatible AMP controllers are available.

If RFCOMM-over-AMP is used, channels that have not been involved in a move procedure and have not enabled any of the automatic move flags may still receive this notification. This is a consequence of the RFCOMM protocol design where only one L2CAP channel is used – and so, only one RFCOMM channel gets to choose whether to allow or deny channel move procedures.

If this is enabled *and* the CSR_BT_AMPM_EVENTS_AUTO_MOVE event mask flag was also set, the AMPM will inform the application about automatic move channel procedure results. Figure 15 shows the MSC for this scenario:

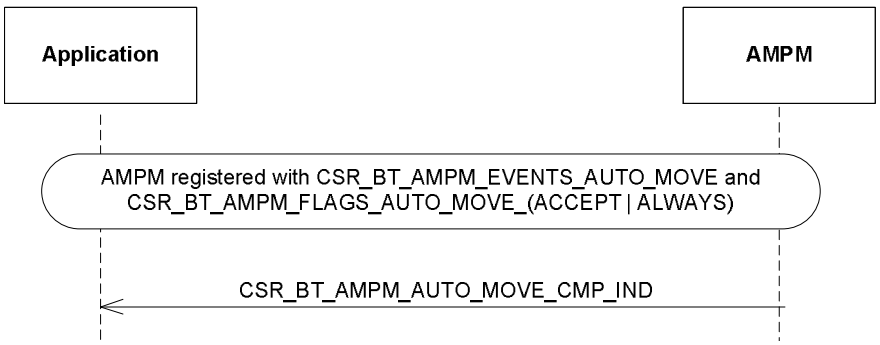


Figure 15: Automatic Move Result

The process is entirely automatic, and the application does not respond to the CSR_BT_AMPM_AUTO_MOVE_CMP_IND. The members in the CSR_BT_AMPM_AUTO_MOVE_CMP_IND are explained in Table 13.

Type	Member	Description
CsrBtAmpmPrim	Type	Signal identity – always set to CSR_BT_AMPM_AUTO_MOVE_CMP_IND

Type	Member	Description
CsrBtConnId	btConnId	Global Bluetooth connection identifier of the channel, which the peer wants to move.
CsrBtAmpController	localId	Local AMP controller ID, which the peer wants to move the channel to.
CsrBtResultCode	resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier = CSR_BT_SUPPLIER_AMPM then the possible result codes can be found in csr_bt_ampm_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.
CsrBtSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 16: Members in a CSR_BT_AMPM_AUTO_MOVE_CMP_IND primitive

Note: The `CSR_BT_AMPM_AUTO_MOVE_CMP_IND` and `CSR_BT_AMPM_MOVE_CFM` primitive types are the same – only the signal identifiers (the *type* member) are different.

4 Document References

Document	Reference
Bluetooth® Core Specification 3.0+HS. Available for download at www.bluetooth.org	[1]

Terms and Definitions

AMP	Alternate MAC (media access control) and PHY (physical)
AMPM	AMP Manager
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
BR/EDR	Basic Rate / Enhanced Data Rate. Common name for the standard Bluetooth® radio technology.
CSR	Cambridge Silicon Radio
DUN	Dial Up Networking
FCS	Frame Check Sequence, 16bit CRC used in L2CAP
FTC	File Transfer Client
FTP	File Transfer Protocol
FTS	File Transfer Server
GOEP	Generic Object Exchange Protocol
HCI	Host Controller Interface
L2CAP	Logical Link Control and Adaption Protocol
MSC	Message Sequence Chart
OBEX	Object Exchange Protocol
UniFi™	Group term for CSR's range of chips designed to meet IEEE 802.11 standards

Document History

Revision	Date	History
1	26 SEP 11	Ready for release 18.2.0
2	23 JAN 12	Ready for release 18.2.2

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