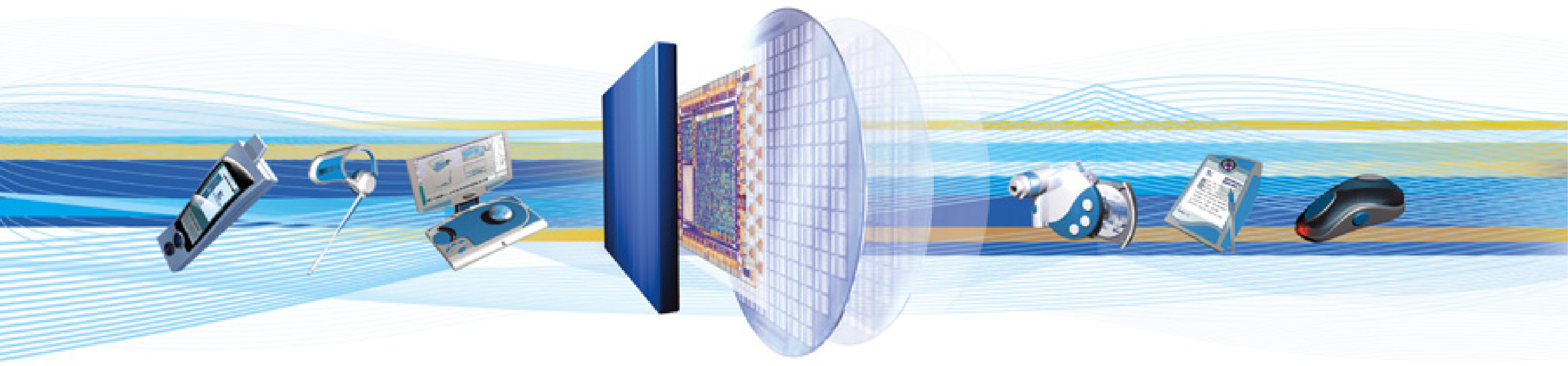
CSR Synergy®

CSR Synergy Framework 3.1.0

AM – Audio Manager

API Description

August 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 used for setting up audio paths in the BlueCore® chip.

The purpose of this document is to describe an easy-to-use interface that will allow application developers to establish and/or release audio connections. The document is intended for SW application developers.

2 Description

2.1 Introduction

The Audio Manager (AM) module is used to request establishment and/or release of audio paths in the BlueCore® chip for chips of the BC7 family or newer. The AM provides a level of abstraction that covers the set of commands provided by the BlueCore® chip in order to allow applications running on top of it to setup the audio paths needed to route the audio data requested.

One command issued to the AM module will map into one or more commands to the BlueCore® chip, to achieve the audio functionality requested by the application.

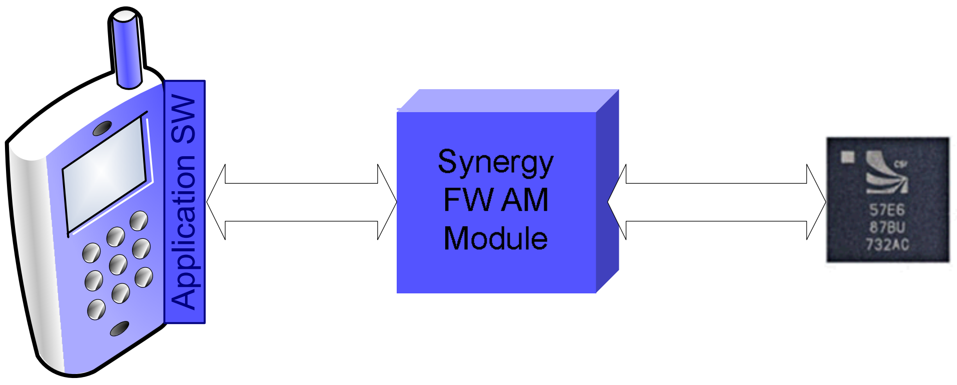
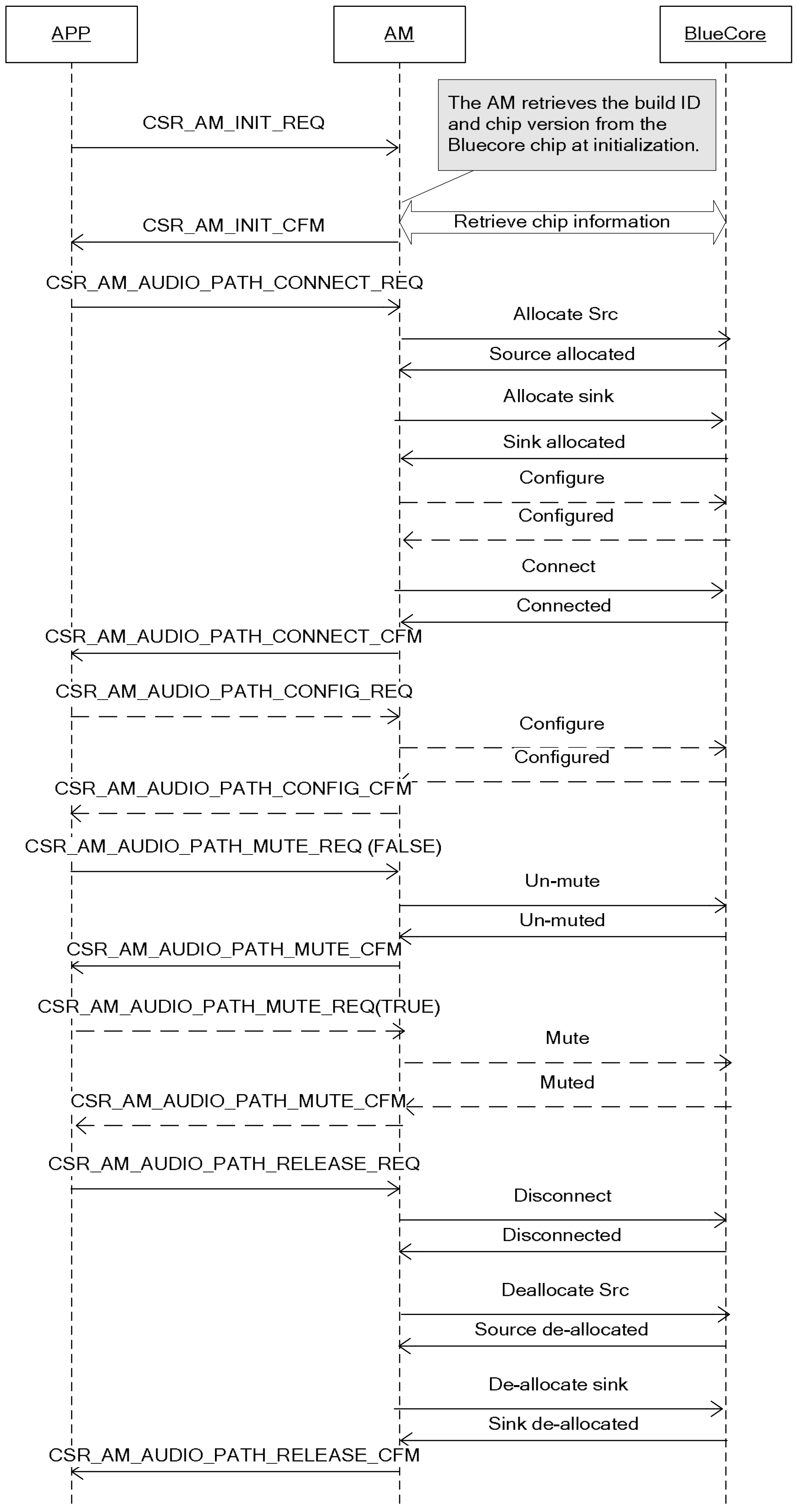


Figure 1: Typical interface.

A common way of using the AM module is depicted in Figure 2. In this figure it is shown that an application needs to initialize the AM before being able to use it. The application does this by sending a CSR_AM_INIT_REQ to the AM module and after it receives a successful CSR_AM_INIT_CFM back, it will be able to set up audio connections through the AM. It also shows what steps are needed as a minimum in order to open an audio connection.



APP
AM
BlueCore
The AM retrieves the build ID and chip version from the Bluecore chip at initialization.
CSR_AM_INIT_REQ
CSR_AM_INIT_CFM
Retrieve chip information
CSR_AM_AUDIO_PATH_CONNECT_REQ
Allocate Src
Source allocated
Allocate sink
Sink allocated
Configure
Configured
Connect
Connected
CSR_AM_AUDIO_PATH_CONNECT_CFM
CSR_AM_AUDIO_PATH_CONFIG_REQ
Configure
Configured
CSR_AM_AUDIO_PATH_CONFIG_CFM
CSR_AM_AUDIO_PATH_MUTE_REQ (FALSE)
Un-mute
Un-muted
CSR_AM_AUDIO_PATH_MUTE_CFM
CSR_AM_AUDIO_PATH_MUTE_REQ(TRUE)
Mute
Muted
CSR_AM_AUDIO_PATH_MUTE_CFM
CSR_AM_AUDIO_PATH_RELEASE_REQ
Disconnect
Disconnected
Deallocate Src
Source de-allocated
De-allocate sink
Sink de-allocated
CSR_AM_AUDIO_PATH_RELEASE_CFM

Figure 2: Example of how the CSR Synergy Framework AM module may be used

2.2 Sequence Overview

The AM module (after the scheduler has initialised it) can do the following operations if the application demands it:

- Establish audio paths
- Add and remove audio endpoints to a connection
- Configure audio connections
- Close audio paths
- Map audio paths for use for Bluetooth® audio

However, it also ensures that common audio resources cannot be used simultaneously for different purposes that are mutually exclusive, i.e. receive FM audio and carry Bluetooth® audio at the same time.

For further details on how to request these jobs, please refer to section 3.

2.3 Intended use

The audio endpoints needed change dynamically, maybe even from connection to connection. Therefore, the application may want to establish the connection only when it needs to provide audio. This approach allows for a more efficient use of the memory resources, as the endpoints only occupy space when actually needed. The application will need to establish the audio, un-mute it, release it and establish it again if needed at a later stage.

3 Interface Description

In this section, a series of MSCs explain the usage of the AM module. The subsections of this chapter also describe the primitives and the functions available to the application.

If CSR_BLUECORE_ONOFF is defined and the BlueCore enters the deactivating state, any outstanding requests will be discarded, and any requests received during the deactivating state will be discarded and the requesting task will not receive a confirm message. Requests received during the deactivated or activating state will be queued and processed when the BlueCore enters the activated state.

3.1 Initialization

If an application wants to make use of the AM module, it needs to initialize it first. The application cannot be sure that the AM module is ready for use before it receives an initialization confirmation from the AM module. At initialization, the AM module retrieves needed information from the BlueCore® chip in order to determine what interface to use towards the BlueCore®. The AM will return an error result if the BlueCore® interface is older than BC7 meaning that the application cannot use the AM to request audio operations.

The application shall use the following function to request the initialization of the AM module:

```
    CsrAmInitReqSend(CsrSchedQid pHandle)
```

Table 1 provides a description of the arguments of that function.

Type	Argument	Description
CsrSchedQid	pHandle	Handle to application, i.e. the queue on which to deliver the CSR_AM_INIT_CFM.

Table 1: Arguments to `CsrAmInitReqSend` function

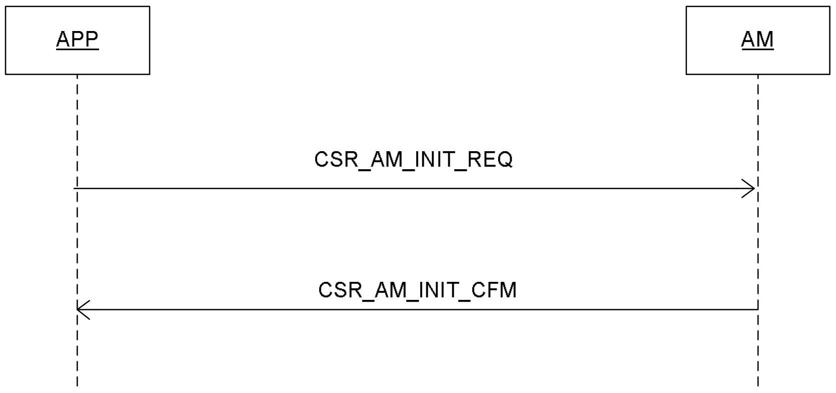


Figure 3: AM initialization

When the AM module has finished handling the CSR_AM_INIT_REQ, it will confirm the request by sending a CSR_AM_INIT_CFM primitive, which contains the parameters described in Table 2 below.

Type	Parameter	Description
CsrAmPrim	type	Signal identity – always CSR_AM_INIT_CFM.
CsrResult	result	Result code. Currently this can only be - CSR_RESULT_SUCCESS (0x0000) or - CSR_AM_RESULT_UNSUPPORTED (0x0001)

Type	Parameter	Description
CsrUint16	buildId	Build ID read from the firmware. This helps the application decide whether the AM shall be used or audio needs to be established directly towards the firmware.

Table 2: Parameters in a CSR_AM_INIT_CFM primitive

Once the AM is active, after it returns success in the CSR_AM_INIT_CFM, the application shall request audio operations through the AM and not directly to the chip to avoid malfunctioning.

3.2 Establishing an audio path

An audio connection consists of at least two endpoints: a source to provide the audio data and a sink to receive the audio data. One sink cannot receive audio data from more than one source; however, a source can deliver audio data to more than one sink.

The application may request to establish an audio connection at any time once the AM is successfully initialized. To do so, it shall use the function, which will issue the CSR_AM_AUDIO_PATH_CONNECT_REQ primitive to the audio manager:

```
   CsrAmAudioPathConnectReqSend(CsrSchedQid pHandle, CsrEndpointType source,
CsrEndpointType sourceR, CsrAmSinksToAddType sinks, CsrAudioType audioType, CsrBool
stereo)
```

This interface allows to connect one source with a number of sinks and to configure each of the endpoints before the connection is actually established.

The following structures are needed to define the primitive contents: "CsrEndpointType", "CsrAmEndPointDefType", "CsrAmSinkConfigType" and CsrAmSinksToAddType. The tables below define these structures.

"CsrAmEndPointDefType" uniquely defines an audio endpoint and consists of:

Type	Argument	Description
CsrUint16	endpoint	Identifier of the audio source. Possible values are: - CSR_AM_ENDPOINT_PCM (0x0001) - CSR_AM_ENDPOINT _I2S (0x0002) - CSR_AM_ENDPOINT _CODEC (0x0003) - CSR_AM_ENDPOINT_FM (0x0004) - CSR_AM_NO_ENDPOINT (0xFFFF) All other values are reserved and may be added at a later stage.
CsrUint16	instance	Instance of the endpoint type above. Set to 0xFFFF for automatic selection.
CsrUint16	channel	Channel or slot within the instance specified. Set to 0xFFFF for automatic selection.

"CsrAmSinkConfigType" describes a configuration option and value for a determined endpoint (a sink) referred to by an index. This type must only be used in connection with the "CsrAmEndPointDefType" above and consists of:

Type	Argument	Description
CsrUint8	sinkIndex	The sink index it refers to, of the list of sink definitions provided in the primitive.
CsrUint32	key	The configuration key to change.

Type	Argument	Description
CsrUint32	value	Value that the mentioned key shall take.

"CsrAmSinksToAddType" describes the sink endpoint or endpoints to add and/or connect to in an audio connection. It consists of the following fields:

Type	Argument	Description
CsrUint8	sinkCount	Number of sink entries in the 'newSinkId' field below
CsrAmEndPointDefType	*newSinkId	List of sink endpoint definitions.
CsrUint8	sinkConfigCount	Number of sink configuration elements
CsrAmSinkConfigType	*sinkConfig	Sink configuration elements containing sink index, key and value

The table below provides a description of the "CsrEndpointType" structure type used in the function.

Type	Argument	Description
`CsrAmEndPointDefType`	endpoint	Type of endpoint as described above.
CsrUint8	configDataCount	Number of configuration element pairs in data field below.
CsrUint32	*configData	This field can contain several configuration parameters. The data format to use is TV (Type-Value) for each configuration element. Both the type and value fields are 32-bit long each. If no configuration data is present, default values will apply. For more information about audio connection configuration, refer to chapter 3.4.

The table below provides a description of the arguments of that function.

Type	Argument	Description
CsrSchedQid	pHandle	Handle to application, i.e. the queue on which to deliver the CSR_AM_AUDIO_PATH_CONNECT_CFM.
CsrEndpointType	source	Structure as defined in the table above. For stereo audio, this field will designate the left side.
CsrEndpointType	sourceR	Structure as defined in the table above. Only needed for stereo audio, to designate the right side. Not needed for mono audio or if the endpoint is of the same type as the left side (i.e. FM or codec). When this field is not needed for the connection, the application shall fill it accordingly (CSR_AM_NO_ENDPOINT).
CsrAmSinksToAddType	sinks	Structure with the sinks to add
CsrAudioType	audioType	FM, WBS, AURISTREAM or CVSD
CsrBool	stereo	If TRUE, need to allocate both right and left channels for each endpoint.

Table 3: Arguments to `CsrAmAudioPathConnectReqSend` function

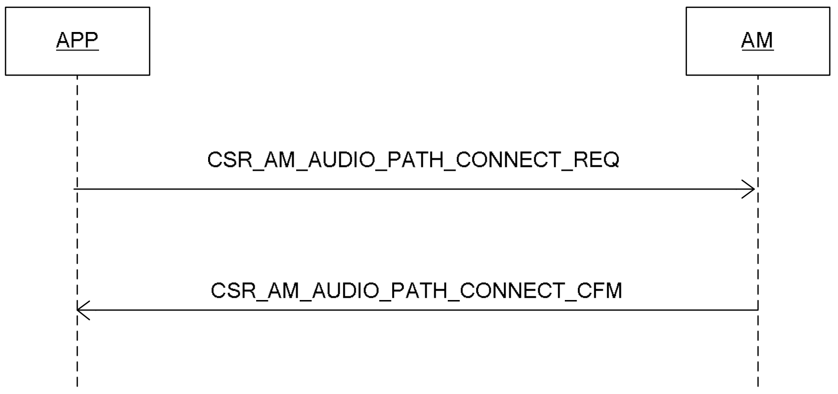


Figure 4: AM audio connection establishment

The AM will issue a CSR_AM_AUDIO_PATH_CONNECT_CFM message when the establishment operation ends. The AM will automatically allocate the endpoints requested and link them together if possible. The operation may fail for different reasons, in which case the application will received an error response. The CSR_AM_AUDIO_PATH_CONNECT_CFM message contains a result code and a connection identifier that the application shall use thereafter to request operations related to the connection established. The table below shows the contents of the primitive CSR_AM_AUDIO_PATH_CONNECT_CFM.

Type	**Parameter**	**Description**
CsrAmPrim	type	Signal identity – always CSR_AM_AUDIO_PATH_CONNECT_CFM.
CsrResult	result	Result code. This can be - CSR_RESULT_SUCCESS (0x0000) - CSR_AM_RESULT_UNSUPPORTED (0x0001) - CSR_AM_RESULT_ERROR_SINK_BUSY (0x0002) - CSR_AM_RESULT_ERROR_SRC_BUSY (0x0003) - CSR_AM_RESULT_INVALID_PARAMETER (0x0008) - CSR_RESULT_FAILURE (0xFFFF)
CsrUint16	amConnectionId	Identifier of the connection.

Table 4: Parameters in a CSR_AM_AUDIO_PATH_CONNECT_CFM primitive

3.3 Adding or removing audio sinks

Once an audio connection is established, it is still possible to add a new sink to it, or remove a sink from it. In case of adding a sink, if the sink requested cannot be added because it is already in use in another connection, the AM will ensure that the connection remains untouched and that the application receives a proper indication. Removing a sink from a connection when it is the only sink present is equivalent to releasing the audio connection. This will provoke not only that the information about the sink will be lost, but also that the information about the source will be lost. In that case, the AM will issue a CSR_AM_AUDIO_PATH_RELEASE_IND message (see chapter 3.5).

To add a sink to a connection, the application shall call the function:

```
   CsrAmAudioPathAddSinkReqSend(CsrSchedQid pHandle, CsrUint16 connectionId,
CsrAmSinksToAddType sinks)
```

This will provoke the creation of a new sink endpoint that will be aliased to the existing sink or sinks in the connection indicated. The new sink will inherit the configuration already applied to the existing sink or sinks in the connection. The table below describes the parameters of the function.

Type	Argument	Description
CsrSchedQid	pHandle	Handle to application, i.e. the queue on which to deliver the CSR_AM_AUDIO_PATH_ADD_SINK_CFM.
CsrUint16	amConnectionId	Identifier of the connection received in the CSR_AM_AUDIO_PATH_CONNECT_CFM
`CsrAmSinksToAddType`	sinks	Structure with the sinks to add

Table 5: Arguments to `CsrAmAudioPathAddSinkReqSend` function

The answer to this is a CSR_AM_AUDIO_PATH_ADD_SINK_CFM primitive described in the table below.

Type	Argument	Description
CsrAmPrim	type	Signal identity – always CSR_AM_AUDIO_PATH_ADD_SINK_CFM.
CsrResult	result	Result code. This can be - CSR_RESULT_SUCCESS (0x0000) - CSR_AM_RESULT_ERROR_SINK_BUSY (0x0002) - CSR_AM_RESULT_WRONG_SINK (0x0006) - CSR_AM_RESULT_UNKNOWN_CONNECTION (0x0007) - CSR_RESULT_FAILURE (0xFFFF)
CsrUint16	amConnectionId	Identifier of the connection received in the CSR_AM_AUDIO_PATH_ADD_SINK_REQ

Table 6: Parameters in a CSR_AM_AUDIO_PATH_ADD_SINK_CFM primitive

To remove a sink from a connection, the application shall call the function:

```
   CsrAmAudioPathRemoveSinkReqSend(CsrSchedQid pHandle, CsrUint16 amConnectionId,
CsrAmEndPointDefType sinkId)
```

To following table describes the parameters of this function.

Type	Argument	Description
CsrSchedQid	pHandle	Handle to application, i.e. the queue on which to deliver the CSR_AM_AUDIO_PATH_REMOVE_SINK_CFM.
CsrUint16	amConnectionId	Identifier of the connection received in the CSR_AM_AUDIO_PATH_CONNECT_CFM
CsrAmEndPointDefType	sinkId	Audio sink endpoint to remove.

Table 7: Arguments to `CsrAmAudioPathRemoveSinkReqSend` function

When the operation completes the AM shall issue the CSR_AM_AUDIO_PATH_REMOVE_SINK_CFM message with the following parameters:

Type	Argument	Description
CsrAmPrim	type	Signal identity – always CSR_AM_AUDIO_PATH_REMOVE_SINK_CFM.

Type	Argument	Description
CsrResult	result	Result code. This can be - CSR_RESULT_SUCCESS (0x0000) - CSR_AM_RESULT_WRONG_SINK (0x0006) - CSR_AM_RESULT_UNKNOWN_CONNECTION (0x0007) - CSR_RESULT_FAILURE (0xFFFF)
CsrUint16	amConnectionId	Identifier of the connection received in the CSR_AM_AUDIO_PATH_REMOVE_SINK_REQ

Table 8: Parameters in a CSR_AM_AUDIO_PATH_REMOVE_SINK_CFM primitive

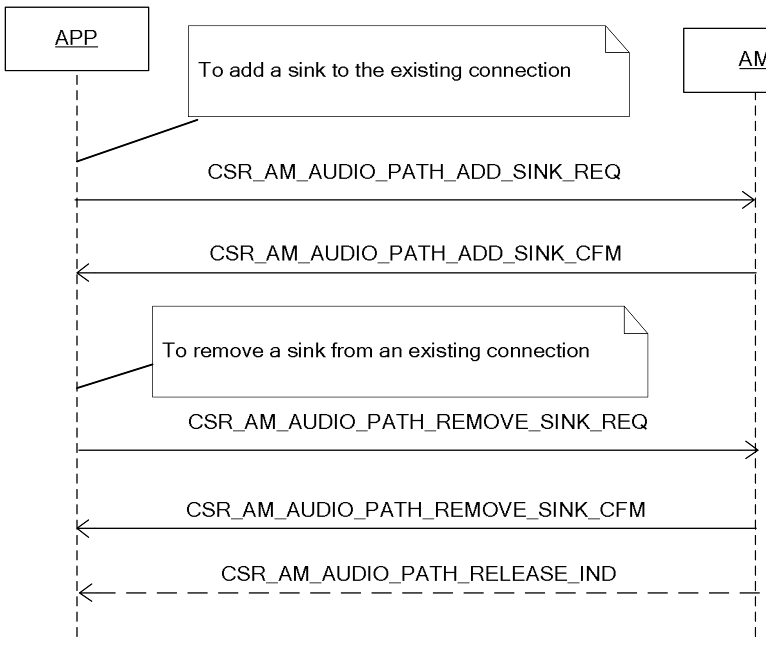

APP
To add a sink to the existing connection
AM
CSR_AM_AUDIO_PATH_ADD_SINK_REQ
CSR_AM_AUDIO_PATH_ADD_SINK_CFM
To remove a sink from an existing connection
CSR_AM_AUDIO_PATH_REMOVE_SINK_REQ
CSR_AM_AUDIO_PATH_REMOVE_SINK_CFM
CSR_AM_AUDIO_PATH_RELEASE_IND

Figure 5: AM adding and removing a sink to a connection

3.4 Configuring an audio connection

Some characteristics of an audio connection can be changed dynamically after connection establishment. The application may do so by issuing the CSR_AM_AUDIO_PATH_CONFIG_REQ message, indicating the connection to configure, the characteristic to change and the new value to apply. The characteristics that the application can configure may vary depending on the chip and firmware used. Configurable characteristics include gain and sample rate.

To configure an audio endpoint the application shall use the function:

```
   CsrAmAudioPathConfigReqSend(CsrSchedQid pHandle, CsrUint16 amConnectionId,
CsrAmEndpointType epType, CsrUint16 sinkId, CsrUint16 configDataLen, CsrUint8
*configData)
```

The table below provides a description of the arguments of the function.

Type	Argument	Description
CsrSchedQid	pHandle	Handle to application, i.e. the queue on which to deliver the CSR_AM_AUDIO_PATH_CONFIG_CFM.

Type	Argument	Description
CsrUint16	amConnectionId	Identifier of the connection provided in the CSR_AM_AUDIO_PATH_CONNECT_CFM primitive described in 3.2
CsrAmEndpointType	epType	This field indicates what endpoint or endpoints of the connection to configure. It can take the values: - CSR_AM_EP_SOURCE_LEFT (0x0000) - CSR_AM_EP_SOURCE_RIGHT (0x0001) - CSR_AM_EP_SOURCE_ALL (0x0002) - CSR_AM_EP_SINK_LEFT (0x0003) - CSR_AM_EP_SINK_RIGHT(0x0004) - CSR_AM_EP_SINK_BOTH_SIDES (0x0005) - CSR_AM_EP_ALL_LEFT_SINKS (0x0006) - CSR_AM_EP_ALL_RIGHT_SINKS (0x0007) - CSR_AM_EP_ALL_SINKS_BOTH_SIDES (0x0008) - CSR_AM_EP_ALL_LEFT (0x0009) - CSR_AM_EP_ALL_RIGHT (0x000A) - CSR_AM_EP_ALL_BOTH_SIDES (0x000B)
CsrUint16	sinkId	This field is only relevant if the “epType” is CSR_AM_EP_SINK_LEFT, CSR_AM_EP_SINK_RIGHT or CSR_AM_EP_SINK_BOTH_SIDES and it determines which of the sinks to configure. Possible values are: - CSR_AM_ENDPOINT_PCM (0x0001) - CSR_AM_ENDPOINT _I2S (0x0002) - CSR_AM_ENDPOINT _CODEC (0x0003) - CSR_AM_ENDPOINT_FM (0x0004) All other values are reserved and may be added at a later stage
CsrUint16	configDataLen	Number of configuration element pairs in data field below.
CsrUint8	*configData	This field can contain several configuration parameters. The data format to use is TV (Type-Value) for each configuration element. Both the type and the value field are 32-bit long. If no configuration data is present, default values will apply. The file csr_am_prim.h contains an enumeration with the configurable characteristics.

Table 9: Arguments to `CsrAmAudioPathConfigReqSend` function

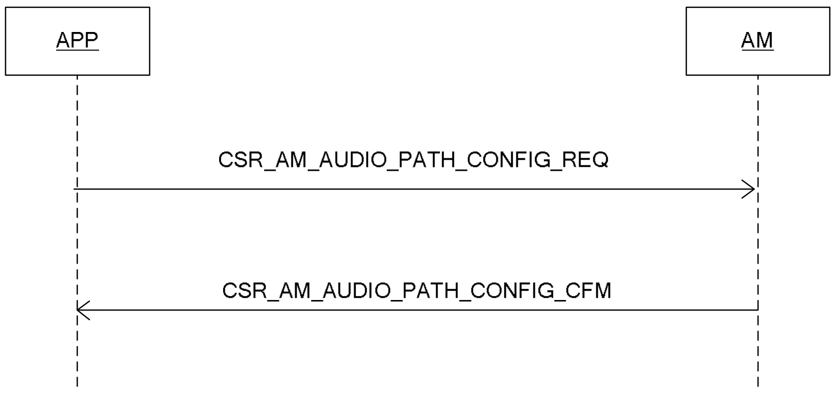


Figure 6: AM audio endpoint configuration

The AM will issue a CSR_AM_AUDIO_PATH_CONFIG_CFM message when the configuration operation ends. Each of the configuration values provided by the application need to be handled separately. Thus, there is a possibility that some of them will succeed while some others fail. The confirmation message will contain only one result code and a connection identifier. If just one configuration value cannot be set, the whole command will be considered as unsuccessful. The parameters of this primitive are:

Type	Parameter	Description
CsrAmPrim	type	Signal identity – always CSR_AM_AUDIO_PATH_CONFIG_CFM.
CsrResult	result	Result code. This can be - CSR_RESULT_SUCCESS (0x0000) - CSR_AM_RESULT_CONFIG_INVALID (0x0004) - CSR_AM_RESULT_UNKNOWN_ID (0x0005) - CSR_RESULT_FAILURE (0xFFFF) All other result codes are reserved for future use and will indicate an error.
CsrUint16	amConnectionId	Identifier of the connection from the CSR_AM_AUDIO_PATH_CONFIG_REQ.

Table 10: Parameters in a CSR_AM_AUDIO_PATH_CONFIG_CFM primitive

3.5 Releasing an audio path

When the application decides to release an audio connection, the AM frees the source and sink or sinks used in that connection. All information about the source and sink or sinks of the connection will be lost and the endpoints will be available for future connections. Beware that any configured characteristics for the connection will be lost as well. A new connection established with the same endpoints at a later stage will need configuring again.

The application shall request this operation once for each connection established before closing down, in order to make sure that all memory allocated in connection with the audio establishment becomes available again. The application shall call the function:

```
    CsrAmAudioPathReleaseReqSend(CsrSchedQid pHandle, CsrUint16 amConnectionId)
```

The parameters needed are:

Type	Argument	Description
CsrSchedQid	pHandle	Handle to application, i.e. the queue on which to deliver the CSR_AM_AUDIO_PATH_RELEASE_CFM.

Type	Argument	Description
CsrUint16	amConnectionId	Identifier of the connection received in the CSR_AM_AUDIO_PATH_CONNECT_CFM message.

Table 11: Arguments to `CsrAmAudioPathReleaseReqSend` function

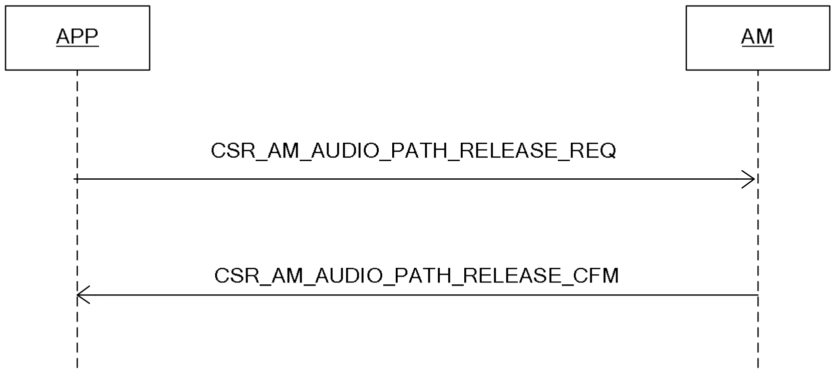


Figure 7: AM release connection

The AM will issue a CSR_AM_AUDIO_PATH_RELEASE_CFM message when the connection is released or the operation fails. The table below shows the parameters of this primitive.

Type	Parameter	Description
CsrAmPrim	type	Signal identity – always CSR_AM_AUDIO_PATH_RELEASE_CFM.
CsrResult	result	Result code. This can be - CSR_RESULT_SUCCESS (0x0000) - CSR_AM_RESULT_UNSUPPORTED (0x0001) - CSR_AM_RESULT_UNKNOWN_CONNECTION (0x0007) - CSR_RESULT_FAILURE (0xFFFF)
CsrUint16	amConnectionId	Identifier of the connection given in the CSR_AM_AUDIO_PATH_RELEASE_REQ message.

Table 12: Parameters in a CSR_AM_AUDIO_PATH_RELEASE_CFM primitive

The audio path will close if the application removes the last sink of the connection. In that case, the AM will issue a CSR_AM_AUDIO_PATH_RELEASE_IND message to the application containing the parameters described in the table below.

Type	Parameter	Description
CsrAmPrim	type	Signal identity – always CSR_AM_AUDIO_PATH_RELEASE_IND.
CsrUint16	amConnectionId	Identifier of the connection given in the CSR_AM_AUDIO_PATH_RELEASE_REQ message.

Table 13: Parameters in a CSR_AM_AUDIO_PATH_RELEASE_IND primitive

3.6 Mapping the audio path

This is relevant for Bluetooth® connections. Once the audio path has been established and connected, before it can be used for Bluetooth® audio, the Bluetooth® shall be mapped to the stream indicating how to code and decode the data (i.e. wide-band speech, CVSD audio, etc…)

The application shall request this operation once for each connection established **before** using it for Bluetooth® audio. The application shall call the function:

```
   CsrAmAudioPathMapScoReqSend(CsrSchedQid pHandle, CsrUint16 amConnectionId,
CsrUint8 mapping)
```

The parameters needed are:

Type	Argument	Description
CsrSchedQid	pHandle	Handle to application, i.e. the queue on which to deliver the CSR_AM_AUDIO_PATH_MAP_SCO_CFM.
CsrUint16	amConnectionId	Identifier of the connection received in the CSR_AM_AUDIO_PATH_CONNECT_CFM message.
CsrUint8	mapping	Type of decoding. It can be: - CSR_AM_WBS_CVSD (0x01) - CSR_AM_WBS_MSBC (0x02)

Table 14: Arguments to `CsrAmAudioPathMapScoReqSend` function

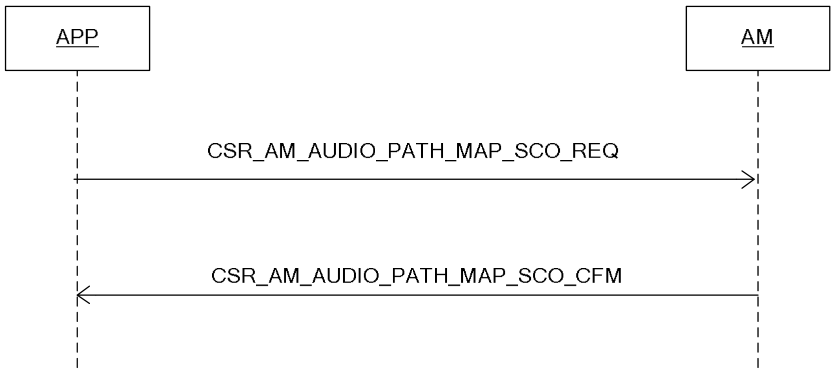


Figure 8: AM map sco connection

The AM will issue a CSR_AM_AUDIO_PATH_MAP_SCO_CFM message when the operation is finished. The table below shows the parameters of this primitive.

Type	Parameter	Description
CsrAmPrim	type	Signal identity – always CSR_AM_AUDIO_PATH_MAP_SCO_CFM.
CsrResult	result	Result code. This can be - CSR_RESULT_SUCCESS (0x0000) - CSR_AM_RESULT_UNSUPPORTED (0x0001) - CSR_AM_RESULT_UNKNOWN_CONNECTION (0x0007) - CSR_RESULT_FAILURE (0xFFFF)
CsrUint16	amConnectionId	Identifier of the connection given in the CSR_AM_AUDIO_PATH_MAP_SCO_REQ message.

Table 15: Parameters in a CSR_AM_AUDIO_PATH_MAP_SCO_CFM primitive

4 Document References

Terms and Definitions

AM	Audio Manager
API	Application Programming Interface
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
FM	Frequency Modulation
MSC	Message Sequence Chart
SW	Software

Document History

Revision	Date	History
1	2 June 10	1st draft. Ready for review
2	3 June 10	2nd draft. Ready for review
3	4 June 10	Changes for consistency in nomenclature after review. Added MSC for mute/un-mute operation. Changed contents of some primitives and format of some fields. Added primitive CSR_AM_AUDIO_PATH_RELEASE_IND
4	14 June 10	Updated after review input received.
5	15 June 10	CsrAmAudioPathConnect changed again. Inconsistencies removed. Parameters organized in a way that there will be no problem with serialization tools.
6	16 Aug. 10	Removed mute/un-mute part as it is not available at the chip yet. Updated to reflect the actual implementation.
7	OCT 2010	Ready for release 2.2.0
8	DEC 2010	Ready for release 3.0.0
9	Aug 2011	Ready for release 3.1.0

TradeMarks, Patents and Licences

Life Support Policy and Use in Safety-critical Compliance

Performance and Conformance

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