BLUETOOTH DOC	Date / Year-Month-Day 2008-06-26	Approved	Revision V10r00	Document No MCAP_SPEC
Prepared Medical Devices WG	e-mail address med-feedback@bluetooth.org			N.B.

MULTI-CHANNEL ADAPTATION PROTOCOL

Abstract

The Multi-Channel Adaptation Protocol (MCAP) is a versatile L2CAP-based protocol that provides a Control Channel to create and manage a plurality of Data Channels.

Revision History

Revision	Date	Comments
D05r09	24 January 2007	Revision 0.5 approved by BARB
D09r05	29 September 2007	Revision 0.9 approved by BARB
D10r14	18 June 2008	Revision 1.0 approved by BARB
D10r15	25 June 2008	Prepare for publication
V10r00	26 June 2008	Adopted by the Bluetooth SIG Board of Directors

Contributors

Company	Name
A & D Medical	Jerry Wang
Bluegiga	Hermanni Suominen
Broadcom	Victor Zhodzishsky
connectBlue	Mats Andersson
Cybercom	Pär Sandell Ulf Karlsson
Ezurio	Nick Hunn
IBM	Michael Nidd
Intel Corporation	Robert D. Hughes (editor) Doug Bogia
MindTree	Dennis Mathews
Motorola	Fabio Grigorjev
Nonin	Jayant Parthasarathy
Philips	Lars Schmitt
Radio, Digital Technique	Brad Tipler
Socket Mobile	Len Ott
Stollmann	Karsten Aalders Juergen Wichmann
Symbian	Tim Howes
Welch Allyn	Jim DelloStritto

Disclaimer and Copyright Notice

The copyright in this specification is owned by the Promoter Members of *Bluetooth*® Special Interest Group (SIG), Inc. ("*Bluetooth* SIG"). Use of these specifications and any related intellectual property (collectively, the "Specification"), is governed by the Promoters Membership Agreement among the Promoter Members and *Bluetooth* SIG (the "Promoters Agreement"), certain membership agreements between *Bluetooth* SIG and its Adopter and Associate Members (the "Membership Agreements") and the *Bluetooth* Specification Early Adopters Agreements (1.2 Early Adopters Agreements) among Early Adopter members of the unincorporated *Bluetooth* SIG and the Promoter Members (the "Early Adopters Agreement"). Certain rights and obligations of the Promoter Members under the Early Adopters Agreements have been assigned to *Bluetooth* SIG by the Promoter Members.

Use of the Specification by anyone who is not a member of *Bluetooth* SIG or a party to an Early Adopters Agreement (each such person or party, a "Member") is prohibited. The legal rights and obligations of each Member are governed by their applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement. No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

Any use of the Specification not in compliance with the terms of the applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement is prohibited and any such prohibited use may result in termination of the applicable Membership Agreement or Early Adopters Agreement and other liability permitted by the applicable agreement or by applicable law to *Bluetooth* SIG or any of its members for patent, copyright and/or trademark infringement.

THE SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

Each Member hereby acknowledges that products equipped with the *Bluetooth* technology ("*Bluetooth* products") may be subject to various regulatory controls under the laws and regulations of various governments worldwide. Such laws and regulatory controls may govern, among other things, the combination, operation, use, implementation and distribution of *Bluetooth* products. Examples of such laws and regulatory controls include, but are not limited to, airline regulatory controls, telecommunications regulations, technology transfer controls and health and safety regulations. Each Member is solely responsible for the compliance by their *Bluetooth* Products with any such laws and regulations and for obtaining any and all required authorizations, permits, or licenses for their *Bluetooth* products related to such regulations within the applicable jurisdictions. Each Member acknowledges that nothing in the Specification provides any information or assistance in connection with securing such compliance, authorizations or licenses**. NOTHING IN THE SPECIFICATION CREATES ANY WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING SUCH LAWS OR REGULATIONS.**

ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS OR FOR NONCOMPLIANCE WITH LAWS, RELATING TO USE OF THE SPECIFICATION IS EXPRESSLY DISCLAIMED. BY USE OF THE SPECIFICATION, EACH MEMBER EXPRESSLY WAIVES ANY CLAIM AGAINST *BLUETOOTH* SIG AND ITS PROMOTER MEMBERS RELATED TO USE OF THE SPECIFICATION.

Bluetooth SIG reserve the right to adopt any changes or alterations to the Specification as it deems necessary or appropriate.

Document Terminology

The Bluetooth SIG has adopted Section 13.1 of the IEEE Standards Style Manual [3], which dictates use of the words ``shall'', ``should'', ``may'', and ``can'' in the development of documentation, as follows:

- The word **shall** is used to indicate mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (shall equals is required to).
- The word **should** is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (should equals is recommended that).
- The word **may** is used to indicate a course of action permissible within the limits of the standard (may equals is permitted).
- The word **can** is used for statements of possibility and capability, whether material, physical, or causal (can equals is able to).
- The use of the word **must** is deprecated and shall not be used when stating mandatory requirements; must is used only to describe unavoidable situations.
- The use of the word **will** is deprecated and shall not be used when stating mandatory requirements; will is only used in statements of fact.

Contents

1 Introduction

1.1 Scope

MCAP defines a simple protocol for communicating with devices as if they were connected over a locally attached cable. This document defines the protocol and generic procedures that can be used by compatible profiles to fulfill a wide variety of usage models.

This protocol enables the following capabilities:

- Provisions for a standardized, structured approach for using a Control Channel to connect and coordinate necessary Data Channels, offering a capability for retaining state references between reconnections. This approach provides structured management of a group of related channels between a given pair of devices which support this protocol and a compatible profile.
- Permits multiple simultaneous data channels.
- Connection-oriented [see Section 3]; which is likely to ensure more reliable behavior when a device moves out of range or disconnects (either inadvertently or intentionally), because the device can recognize the condition and make appropriate decisions.
- Facilitates re-establishment of wireless connections that have been disconnected (intentionally or unintentionally) by associating context identifiers with connections, and allowing success or failure of a reconnect operation to be agreed by the two endpoints.
- Defines a standardized Clock Synchronization Protocol that uses the Control Channel and the shared Bluetooth Master Clock to coordinate local clocks.
- Designed to operate specifically with designated Data Exchange Specifications, which define the base data protocol and command set and device data formats for various supported devices. This facilitates genuine interoperability between data *Source*s and *Sink*s minimizing the need for proprietary drivers on either side to format and interpret the data. The specific Data Exchange Specifications are defined in profiles that reference MCAP.
- Allows profiles which reference MCAP to make use of elements of new L2CAP features [1] such as Enhanced Retransmission Mode, Streaming Mode and optional FCS for improved reliability and flexibility.
- Inexpensive to implement due to small list of relatively simple control commands and low code-space requirements.

1.2 Conformance

If conformance to this protocol is claimed, all capabilities indicated as mandatory for this protocol shall be supported in the specified manner (process mandatory). This also applies for all optional and conditional capabilities for which support is indicated.

All mandatory, optional and conditional capabilities, for which support is indicated, are subject to verification as part of the Bluetooth Qualification Program.

1.3 Symbols and Conventions

1.3.1 Requirement Status Symbols

In this document the following symbols are used:

M = Mandatory to support (used for capabilities that shall be implemented in the protocol)

O = Optional to support (used for capabilities that may be implemented in the protocol)

C = Conditional to support (used for capabilities that shall be implemented in case a certain other capability is supported)

X = Excluded (used for capabilities that may be supported by the unit but shall never be used in the protocol)

N/A = Not Applicable (in the given context it is impossible to use this capability)

Some excluded capabilities (identified as X) are capabilities that, according to the relevant Bluetooth specification, are mandatory. These are features that may degrade operation of devices following this protocol. Therefore, these features shall never be activated while a unit is operating as a unit within this protocol.

1.4 Related Specifications

1.4.1 Core Specification

“Core Specification” refers to the Bluetooth Core Specification 2.0 + EDR [1] or later versions of the Bluetooth Core Specification adopted by the Bluetooth® SIG.

1.4.2 Data Exchange Specifications

A Data Channel carries the application content as described in the Data Exchange Specifications. The Data Exchange Specifications that are used are specified in profiles that reference MCAP. These Data Exchange Specifications generally define a base data protocol and command set (also referred to as Data Exchange Protocol), and device data formats (also referred to as Device Data Specializations) for various supported devices.

2 Protocol Overview

MCAP defines the method by which data connections are established between two devices. Full interoperability depends on compatible application implementations on both devices.

A Control Channel is used to facilitate Data Channel creation by establishing the desired logical endpoint for a new connection, or state reference for re-establishment of an old connection. The Control Channel is also used for synchronization commands that coordinate local clocks on multiple devices. The former of these purposes is managed using the “Standard Op Codes,” and is addressed in this section. The latter is managed using the “Clock Synchronization Protocol Op Codes,” and is presented in Section 5.

2.1 Configurations and Roles

The following terms are used to define the roles and configurations of the protocol:

2.1.1 Source / Sink

Source refers to a *Source* of data defined by the Data Exchange Specifications, and a *Sink* is a receiver for that data. The *Source* may generate that data from sensors, or may relay data actually collected by some other device. The *Sink* may be a display unit, a store-and-forward intermediary, or any other consumer of Data Exchange Specification data. While the difference between *Source* and *Sink* is supported through flags that may act as indicators to the Data Exchange Specifications, *Sources* and Sinks have the same observable behavior at the MCAP level. These two classes of device form a bigraph (i.e. members of each group connect with members of the other, but not each other), which is relevant for testing, but *Source* and *Sink* devices have exactly the same set of MCAP operations available to them, and their responses are similarly indistinguishable.

2.1.2 Control Channel / Data Channel

The Control channel is the first L2CAP channel established between two instances of MCAP. This channel facilitates the creation of Data Channels, over which actual data (as defined in the Data Exchange Specifications) can be exchanged. Data Channels can be bi-directional, and carry data useful to higher layers (see “MCAP Data Link,” defined below). The Control Channel carries commands triggered by higher layers, but does not carry any data other than the commands defined in this protocol specification. When the Control Channel is closed, all of its associated Data Channels are closed as a side-effect. The details of this process are the main subject of Section 4.

The devices are “connected” (with respect to MCAP) when a Control Channel is established between those devices. So long as the Control Channel remains connected, the two devices will remain connected.

The Control Channel is used for session initiation, and to give context to the Data Channel connections (described in Section 4.1.1). For example, the Control Channel is used to identify the logical endpoints to be used by a new Data Channel, and to request a reconnection of a Data Channel (reconnecting to an application context that has already been established).

One or more Data Channels are used to exchange device data as defined by the Data Exchange Specifications.

Control and Data Channel configuration requirements are defined by profiles that reference MCAP.

2.1.3 MCAP Data End Point (MDEP) and MDEP ID

A device that implements MCAP may have one or more logical functions; each function is referred to as an MCAP Data End Point (MDEP). Each MDEP is allocated an identifier (MDEP ID) that is locally unique to that MCAP instance, and can be used to refer uniquely to the corresponding function (as in the creation of an MDL – described below).

Normally, MDEP IDs will be shared via the Service Discovery Protocol (SDP) by a profile using MCAP.

2.1.4 MCAP Data Link (MDL) and MDL ID

A MCAP Data Link (MDL) identifies a set of two MDEPs, and is explicitly created by one of the two participating devices as a result of a request from the Control Channel. Each MDL has an identifier (MDL ID) that is unique for the pair of devices involved. After creating an MDL ID, either of the two participating devices may request a “reconnection” (described in Section 4.1.1) to that MDL ID. If accepted by the device receiving the reconnection request, this indicates that the data layers are aware that it is a reconnection operation, and will be able to re-establish the state of that connection. The purpose of this operation is to allow data layers to establish one-time exchanges of configuration data when the MDL is first created, and be able to bypass this overhead for future exchanges.

An MDL ID is a context reference for a data connection. It is useful only because it associates a Create operation with subsequent Reconnect operations. The use of Reconnect allows configuration exchanges to be optimized by not having to re-exchange already-known data. If a device does not retain state (context) between connections, then it will never need to use the Reconnect operation, so the MDL IDs used in its calls to Create have no significance. The MDL ID provides a "handshake" value for the Reconnect operation: using it in a Reconnect operation indicates that the initiator of the operation remembers a state with this ID, and replying with a "Success" response indicates that the acceptor also remembers this state. If the initiator had lost this state information, it would not have initiated a Reconnect operation. If the acceptor had lost the state information, it would not have accepted the Reconnect operation. In this way, before the Data Channel is opened, both parties already know in which state to start.

The ability to reconnect a Data Channel allows an MCAP implementation to verify that both devices agree upon a context identifier, ensuring the reconnection operation produces a consistent state. This is useful for power conservation or to overcome interruptions in the physical channel.

2.1.5 MCAP Communications Link (MCL)

A MCAP Communications Link (MCL) identifies the full collection of L2CAP connections between two instances of MCAP, comprising a Control Channel and zero[1] or more Data Channels. The Control Channel is always present while the MCL is active.

2.1.6 Initiator / Acceptor

In L2CAP, to establish a communication channel, one device is required to wait to receive connections (operating as a server, having allocated a Protocol/Service Multiplexer [1] (PSM) to identify which incoming requests it is prepared to process). In this document, the device that receives a connection request to establish a particular Control Channel (thereby connecting an MCL) is referred to as the *Acceptor* for discussions relating to that MCL. When another device transmits a request to connect to that PSM, an L2CAP Channel ID is created by the L2CAP implementation, and communication is then possible on the new L2CAP channel. Because the second device triggered the creation of a Channel ID, it is referred to as the *Initiator*. These terms are useful because they help clarify roles in L2CAP connections, while being independent of such things as the physical link and the identity of the Bluetooth master.

Normally, the PSM will be shared via SDP by a profile using MCAP.

2.1.7 Summary of Key Terms

Term	Summary
Abort	Term used to define aborting an attempt to connect to one or more Data Channels using the MD_ABORT_MDL operation
Acceptor	The device which accepted the Control Channel connection from the Initiator
ACL link	An asynchronous (packet-switched) connection between two devices.
Channel	L2CAP connection
Clock Synchronization Protocol Op Codes	Operations on the Control Channel for managing local timer settings.
Connect	Establishment of an L2CAP Control or Data Channel connection

[1] If Standard Op Codes are supported, then zero is typically a transitional state after a Control Channel is established and before a Data Channel is established. The Clock Synchronization Protocol can function normally in the absence of any Data Channels.

Control Channel	L2CAP channel on which all control commands are transmitted. Required for setting up of Data Channels
Create	Term used to define creation of a Data Channel Connection using the MD_CREATE_MDL operation
Data Channel	L2CAP channel which is created by the Control Channel to transport device specific data.
Delete	Deleting one or all MDLs using the MD_DELETE_MDL operation. To delete an MDL is to de-allocate the corresponding MDL ID.
Disconnect	Disconnection of an MCL or an MDL and essentially refers to the closing of associated L2CAP channels.
Initiator	The device which initiated the Control Channel connection to the Acceptor
L2CAP Channel	Logical Link Control and Adaptation Protocol
PSM	Protocol/Service Multiplexer [1]. MCAP uses Dynamic PSMs in the range of 0x1001 and higher.
MCAP	Multi-Channel Adaptation Protocol defined in this document
MCAP Instance	An MCAP instance is the logical entity that provides the capabilities of MCAP protocol to an instance of a profile that refers to MCAP. An MCAP instance can support multiple remote devices simultaneously.
MCL	MCAP Communication Link. This is comprised of a Control channel and zero or more Data Channels. Note that to disconnect an MCL is the same as closing an L2CAP Control Channel and all associated Data Channels.
MDEP	MCAP Data End Point. A logical data end point (*Source* or *Sink* of data)
MDEP ID	MCAP Data End Point Identifier
MDL	MCAP Data Link. Note that to disconnect an MDL is the same as closing an L2CAP Data Channel.
MDL ID	MCAP Data Link Identifier
Operation	The Request/Response transaction for commands such as Create, Reconnect, Abort or Delete.
Reconnect	Reconnection of an MDL using the MD_RECONNECT_MDL operation. Reconnect is not used for the Control Channel since it would be exactly the same as connecting to it for the first time.
Sink	Receiver of application data.
Source	Transmitter of application data. *Source* devices generally have fewer resources (i.e. memory, CPU) than *Sink* devices.
Standard Op Codes	Operations on the Control Channel for managing Data Channels.

Table 2.1: Summary of Key Terms

2.2 Terminology Examples

The figures that follow show examples using the terminology described in the previous section. Throughout this section, the following notation is used:

Multi-Channel Adaptation Protocol

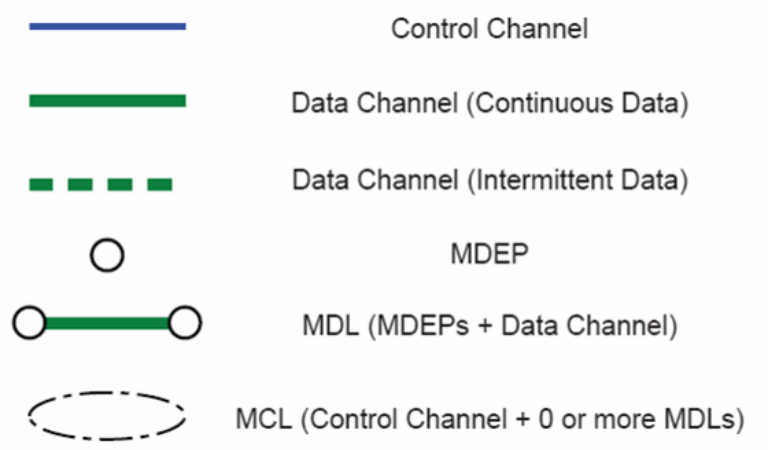

Figure 2.1: Figure Notation

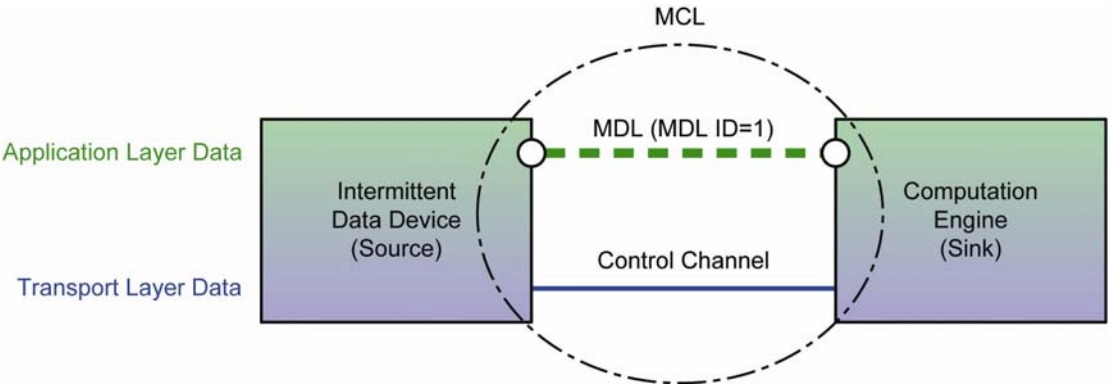

Figure 2.2: Basic Example with Single-function Source Device

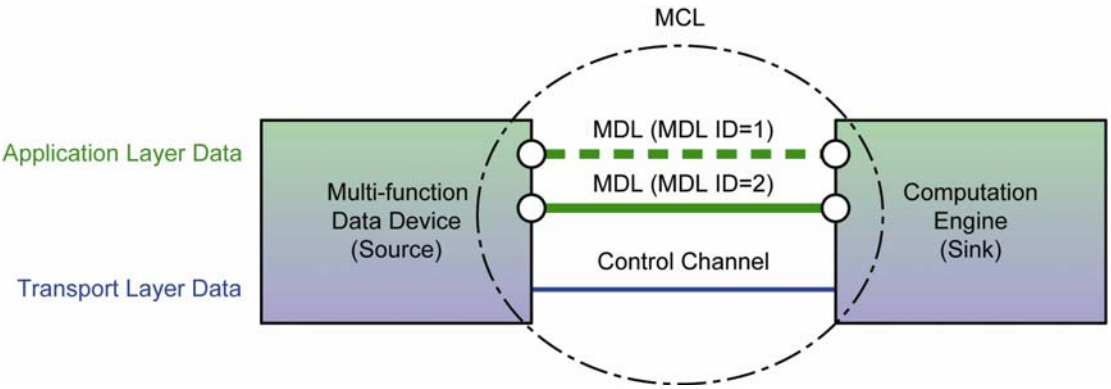

Figure 2.3: Example with Multi-function Source Device

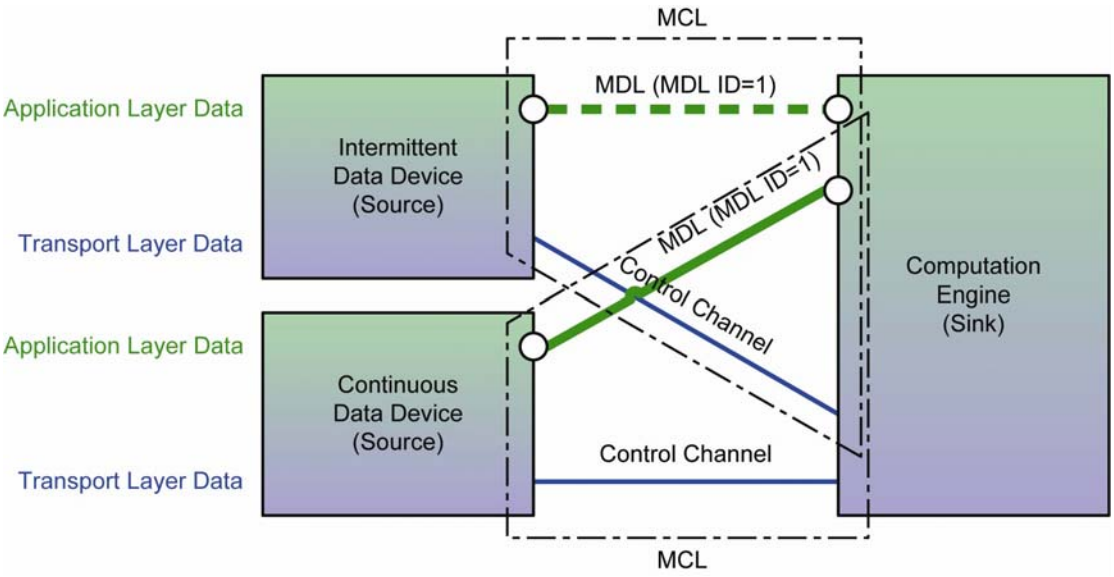

Figure 2.4: Example with Sink running two independent profile instances for Single-function Sources

3 Protocol Layer

- Implementations **shall** be compliant with Bluetooth Core Specification v2.0 + EDR [1] or later versions.

3.1 Transfer Byte Order

- MCAP **shall** transfer data in standard network order (big-endian), which defines more significant (high-order) bytes being transferred before less significant (low-order) bytes, and bit ordering following the same pattern.

As an example, the decimal value 23456 (hex 5BA0) would be encoded (listing from first to last bit transmitted) as 0101101110100000 (i.e. byte 5B followed by byte A0).

3.2 L2CAP Control Channel Configuration

L2CAP Control Channel configuration is specified by profiles that use MCAP.

MCAP is designed for use with a reliable Control Channel and it is necessary that profiles using MCAP configure it for maximum reliability to ensure that all frames are delivered in order, and without duplicates.

- An MCAP instance **shall** support one (and only one) L2CAP Control Channel with each remote device.
- Enhanced Retransmission Mode defined in [5] **should** be used for configuration of the Control Channel.

3.3 L2CAP Data Channel Configuration

L2CAP Data Channel configuration is specified by profiles that use MCAP.

- An MCAP instance supporting standard op codes **shall** support at least one L2CAP Data Channel. Implementations that support only the Clock Synchronization Protocol are not required to support Data Channels.
- Enhanced Retransmission Mode defined in [5] **should** be used for configuration of reliable Data Channels.
- Streaming Mode defined in [5] **should** be used for configuration of streaming Data Channels

3.4 Device Role (Source / Sink) Requirements

Sources and *Sinks* have the same observable behavior at the MCAP level. The role of either *Source* or *Sink* is identified by profiles that reference MCAP typically through the use of a flag in the SDP record.

- *Sources* and Sinks form a bigraph (i.e. members of each group connect with members of the other, but not each other) and therefore **shall** only connect to the opposite member.
- Only one Device Role **shall** be identified per MDEP ID.

– A dual-role device **may** be implemented using two or more MDEP IDs that specify different Device Roles.

4 Control and Data Channels

4.1 Control Channel Protocol

This section defines configuration and creation of the MCL.

The Control Channel carries all MCAP-specific communication which includes both Standard Op Codes and Clock Synchronization Protocol Op Codes. The primary use of Standard Op Codes is to create and reconnect Data Channels (MDLs) within an MCL. Opening or closing the Control Channel is equivalent to connecting or disconnecting the MCL. Clock Synchronization Protocol Op Codes (defined in Section 5) are used for timing synchronization between two or more devices.

- Connections to the Control Channel PSM from a remote device that already has another channel open to the same Control Channel PSM **should** be refused at the L2CAP layer using L2CAP Connection Response “Connection refused – no resources available”.
  - When it is not feasible to refuse the connection directly from L2CAP, such a connection **shall** be closed as quickly as possible, without sending (or responding to) any data traffic.
- Only one Control Channel **shall** exist between two instances of MCAP.

Under certain rare conditions, both devices might simultaneously attempt to initiate a Control Channel connection with each other and may result in two connected Control Channels. If such a condition were to arise, the following procedure is to be followed:

- Each device **shall** disconnect one of the Control Channels which could result in both Control Channels being disconnected.
  - If both Control Channels become disconnected, each device **shall** back off a random amount of time before re-initiating a Control Channel connection.

4.1.1 Protocol Overview

This section provides an informative overview of the Standard Op Codes on the Control Channel protocol. Clock Synchronization Protocol Op Codes are addressed in Section 5. Refer also to key terms defined in Table 2.1.

As indicated in Section 2.1.5, an MCL comprises a Control Channel and zero or more Data Channels between two instances of MCAP. This channel can be said to be active while the underlying L2CAP connection remains available (as established by the underlying Bluetooth implementation).

To Connect or Disconnect an MDL or an MCL refers to the establishment and teardown of the L2CAP connection (a Control Channel in the case of an MCL, and a Data Channel in the case of an MDL) through which that object transports data. Before an MDL can be connected, an MDL ID is required to first be allocated via a Create

operation using the MCL Control Channel. Future connections can Reconnect the MDL by referencing the same MDL ID in their connect operation.

To Delete an MDL is to de-allocate the corresponding MDL ID. A side-effect of deleting an active MDL is to Disconnect that MDL and the related Data Channel, but the primary effect is that future attempts to Reconnect using the deleted MDL ID will fail.

It is important to note that, because each MCL has meaning between exactly two devices, inconsistent states between those devices should never occur. For example, if a Delete operation half-succeeds (device A deletes, and device B does not), it will be discovered in one of two ways: (a) device A sends a Create request for the un-deleted MDL ID on device B, creating a new state that overwrites the old state information, or (b) device B sends a Reconnect request to the deleted MDL ID on device A, receives an invalid MDL response, and knows it needs to tell its data layer to reset its state, and “start from scratch” by sending a fresh Create request.

The L2CAP Data Channel that transports the data for an active MDL, like the L2CAP Control Channel that manages an MCL, is said to be opened or closed when the Bluetooth implementation establishes or tears down (respectively) the L2CAP connection. A Data Channel or Control Channel may be closed for any of several reasons, including intentional causes (i.e. the direct or indirect result of a request by one of the participating applications) and unintentional causes (e.g. range limitations, power failures, etc.)

4.1.2 MCL State Machine

This section is informative. The state machine in this section describes all specified actions, but may not represent all possible implementations. Every MCL state machine acts independently from other MCL state machines instanced in the device. This state machine is the same regardless of the MCL characteristics. A device normally has one MCL state machine instance for each unique Bluetooth device (in an MCAP instance) with which it interacts via MCAP. States and events for a single MCL are described below.

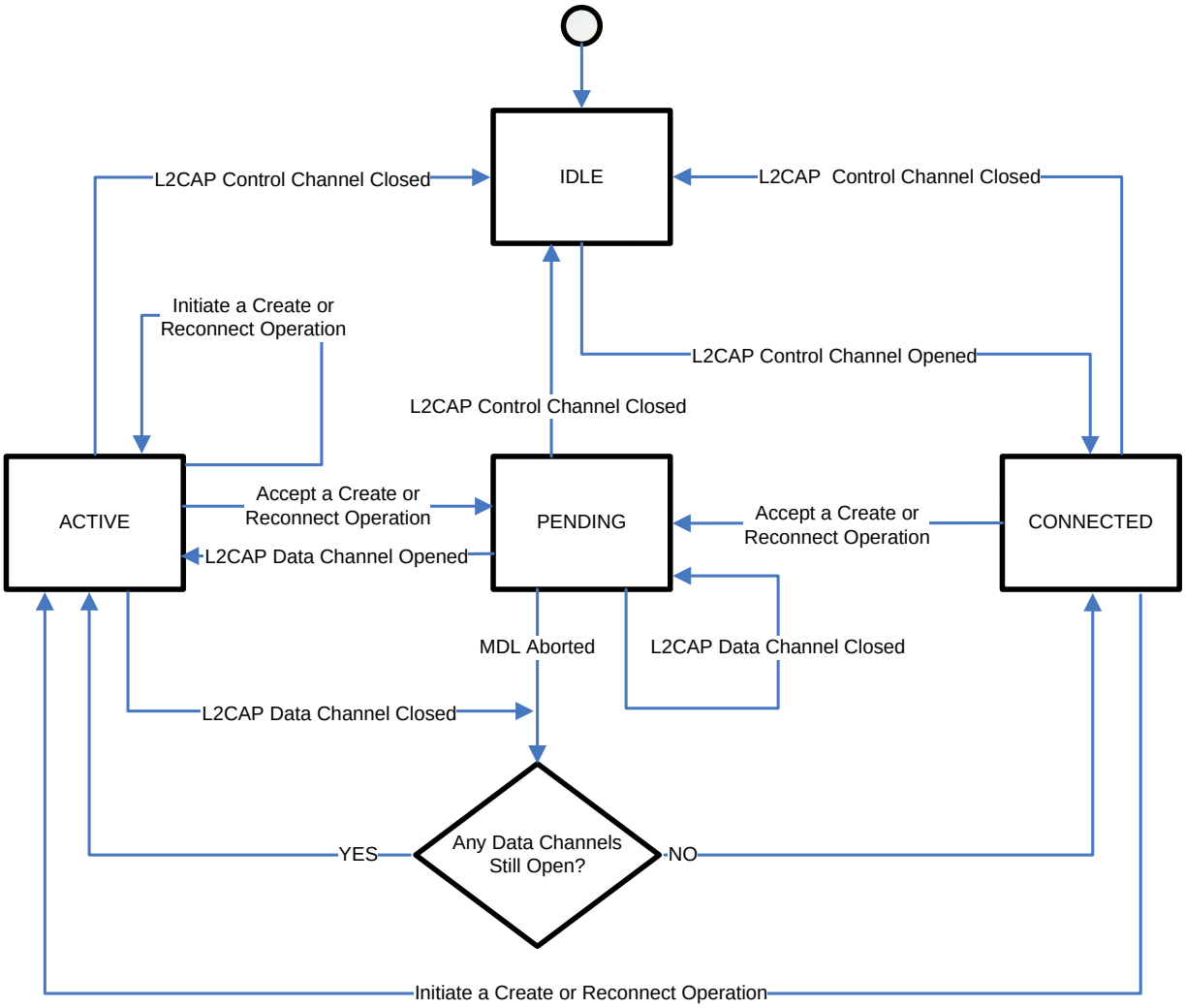

Figure 4.1: MCL State Machine

Short description of the states:

State	Description
IDLE	This MCL is free and available for connection. The only legal transition from this state is the completion of a Control Channel connection.
CONNECTED	MCL connection has been created. The creation of a Data Channel is expected. Legal commands are Create, Reconnect, and Delete
PENDING	A temporary state following the receipt of a Create or Reconnect command during which the device that received the command is waiting to receive a Data Channel connection or reconnection from the device that sent the command. The only legal Standard Op Code (i.e. not including Clock Synchronization Protocol Op Codes) while an MCL is in this state is MD_ABORT_MDL (to give up waiting for the Data Channel to be connected).
ACTIVE	At least one Data Channel is open. Legal commands are Create, Reconnect, and Delete

Table 4.1: Control Channel State Machine - Description of States

- From any of the states for which a Control Channel is open (CONNECTED, ACTIVE, and PENDING) the Control Channel **may** be closed by either party or an external event, such as interference or out of range.
  - When this happens, both sides **shall** ensure that all Data Channels associated with that Control Channel are closed, and MCL state machines on both devices return directly to the IDLE state.
- From any of the states for which a Control Channel is open (CONNECTED, ACTIVE, and PENDING), Clock Synchronization Protocol Op Codes **may** be used as defined in Section 5).
- If a Data Channel connection attempt fails to establish a Data Channel (for example, with a “refused” connection response, or due to any other condition resulting in disconnection before the Data Channel was actually available for data traffic), it **shall** be treated in the state machine (Figure 4.1) as if the connection was opened and then closed.

4.1.3 Overview of Requests and Responses

This section describes the traffic on the Control Channel between two devices implementing Standard Op Codes.

- When a device sends any request it **shall** wait for the corresponding response before sending any other Standard Op Code.
- When a device receives any request, it **shall** send a response before sending any other Standard Op Code on the Control Channel.
- If the Control Channel is closed before a response is received, then the device that had been expecting the response **shall** behave as if it received an Unspecified Error response (refer to Section 4.1.3.4).

- Requests and Responses **shall** be contained in a single L2CAP datagram per request or response.
- If a request is received when a response is expected (i.e. out of sequence operation), then it is assumed that the two devices have sent simultaneous requests. When this occurs the *Initiator* (of the Control Channel) **shall** "win" as indicated by:

    1. The request Op Code received by the *Acceptor* **shall** be processed as normal.
    2. The request Op Code received by the *Initiator* **shall** be ignored.
    3. The next Op Code on the Control Channel **shall** be the response sent from the *Acceptor* to the *Initiator*.
    4. If a device chooses to implement a timeout (after which it assumes the device it is connected to has encountered an internal error, and is unlikely to send a response), then this **shall** be implemented by disconnecting the MCL (by closing the Control Channel and all associated Data Channels), after which it **may** choose to reestablish the MCL connection.

- If a response is received when no request is outstanding, then it **shall** be ignored.
- All messaging requests and responses (as defined in Table 4.2) **shall** be sent on the Control Channel.

4.1.3.1 Request Packet Format

Figure 4.2: Request Packet Format

4.1.3.2 Response Packet Format

The Response Packet format applies to all standard *_RSP codes defined in Table 4.2.

- All responses **shall** include a "Response" value (where zero (0x00) indicates the request was successful).
- The MDL ID sent in a response **shall** be set to the same MDL ID as its corresponding request.

Some responses require an MDL ID although no particular value is appropriate. One example is a 'Response Packet' with Response Code 'Invalid Op Code.' In these cases, the reserved MDL ID 0x0000 is the default value.

- If an invalid Op Code is received (i.e. neither a Standard Op Code nor a Clock Synchronization Protocol Op Code), the Response Packet **shall** set the Response Code to "Invalid Op Code" and the Op Code field to ERROR_RSP.

- If a device does not support Standard Op Codes, but receives a Standard Op Code (see Section 4.3), that device **shall** respond with a "Request Not Supported" Response Code and the Op Code field of the Response Packet **shall** be set to the Response Op Code that corresponds to the unsupported Request Op Code.

  – If a device sends a Standard Op Code and receives a “Request Not Supported” response, the device **shall not** send any further standard op code request operations for the duration of that MCL instance.

- If a device does not claim support for Standard Op Codes, that device **shall not** send any packets that use Standard Op Codes.
- When the Response Code is not Success, the Response Parameters **shall** have length zero.

Figure 4.3: Response Packet Format (Standard Op Code)

Refer to Section 5.2.1 for details regarding the Clock Synchronization Protocol Response Packet format.

4.1.3.3 Op Codes

Table 4.2 shows the valid Op Codes. Note that the Op Code for a Response (RSP) is one number higher than the corresponding Request (REQ) - except ERROR_RSP, which does not have a particular corresponding Request. Where no valid response exists (such as for MD_SYNC_INFO_IND shown in Table 5.1) the Op Code normally allocated for the response, is marked as reserved. The response to a MD_CREATE_MDL_REQ, for example, would have Op Code MD_CREATE_MDL_RSP.

Standard Op Code	Request / Response Name	Description
0x00	ERROR_RSP	Reserved for Invalid Op Code Response Packets
0x01	MD_CREATE_MDL_REQ	Create an MDL, and wait for a connection to the associated Data Channel
0x02	MD_CREATE_MDL_RSP	Response to above request
0x03	MD_RECONNECT_MDL_REQ	Request to prepare to receive a connection on the Data Channel associated with a previously existing MDL
0x04	MD_RECONNECT_MDL_RSP	Response to above request
0x05	MD_ABORT_MDL_REQ	Stop waiting for a Data Channel connection
0x06	MD_ABORT_MDL_RSP	Response to above request
0x07	MD_DELETE_MDL_REQ	Delete an MDL
0x08	MD_DELETE_MDL_RSP	Response to above request
0x09-0x10	Reserved	N/A

0x21-0xFF		
Clock Sync Op Code	**Request / Response Name**	**Description**
0x11-0x20	Refer to section 5.2.	Clock Synchronization Protocol Op Codes

Table 4.2: Request / Response Summary

Note: Standard Op Codes in Table 4.2 are defined throughout this section as applicable. Clock Synchronization Protocol Op Codes are described in Section 5.2.

4.1.3.4 Response Codes

Table 4.3 shows the valid Response Codes.

- The Response Packet **shall** contain the appropriate Response Code.

Response descriptions when using Standard Op Codes are defined in Table 4.3. For response descriptions when using Clock Synchronization Protocol Op Codes, refer to Section 5.2. A list of valid Response Codes is listed in each section for the applicable request.

Response Code	Response Name	Response Code Description
0x00	Success	The corresponding request was received and processed successfully.
0x01	Invalid Op Code	The Op Code received is not valid (i.e. neither a Standard Op Code nor a Clock Synchronization Protocol Op Code).
0x02	Invalid Parameter Value	One or more of the values in the received request is invalid. This shall be used when: - The request length is invalid - Some of the parameters have invalid values and none of the other defined Response Codes are more appropriate.
0x03	Invalid MDEP	The MDEP ID referenced does not exist on this device.
0x04	MDEP Busy	The requested MDEP currently has as many active MDLs as it can manage simultaneously.
0x05	Invalid MDL	The MDL ID referenced is invalid. This shall be used when: - A reserved or invalid value for MDL ID was used. - The MDL ID referenced is not available (was never created, has been deleted, or was otherwise lost), - The MDL ID referenced in the Abort request is not the same value that was used to initiate the PENDING state
0x06	MDL Busy	The device is temporarily unable to complete the request. This is intended for reasons not related to the physical sensor (e.g. communication resources unavailable).
0x07	Invalid Operation	The received request is invalid in the current state. - Abort request was received while not in the PENDING state. - Create, Reconnect, or Delete request was received while in the PENDING state. - A response is received when a request is expected

Response Code	Response Name	Response Code Description
0x08	Resource Unavailable	The device is temporarily unable to complete the request. This is intended for reasons relating to the physical sensor (e.g. hardware fault, low battery), or when processing resources are temporarily committed to other processes.
0x09	Unspecified Error	An internal error other than those listed in this table was encountered while processing the request.
0x0A	Request Not Supported *	The Op Code that was used in this request is not supported.
0x0B	Configuration Rejected	A configuration required by a MD_CREATE_MDL or MD_RECONNECT_MDL operation has been rejected.
0x0C – 0xFF	Reserved	N/A

* Refer to Section 4.1.3.2 (Standard Op Codes) or Section 5.2.3 (Clock Synchronization Protocol Op Codes)

Table 4.3: Response Code Summary

In some situations, an appropriate Response Code does not exist. When that occurs, Unspecified Error is to be used.

- When a device receives an Unspecified Error response, that device **should** close the MCL.

4.1.3.5 MDL IDs

All requests include an MDL ID as a parameter to identify the data connection with which to associate the request. MDL IDs are 16-bit values with the following formats:

MDL ID	Description
0x0000	Reserved
0x0001 - 0xFEFF	Dynamic range
0xFF00 - 0xFFFE	Reserved
0xFFFF	Indicates all MDLs (Used with the MD_DELETE_MDL_REQ only.)

Table 4.4: MDL ID Summary

4.1.3.6 MDEP IDs

MDEP IDs identify the profile Data End-Point with which the Data Channel connection is to be associated. MDEP ID values in the range 0x00 through 0x7F are available for use by profiles using MCAP.

MDEP ID	Description
0x00 - 0x7F	Available for use
0x80 - 0xFF	Reserved

Table 4.5: MDEP ID Summary

4.1.3.7 Requests and Responses

4.1.3.7.1 MD_CREATE_MDL

Figure 4.4: MD_CREATE_MDL_REQ Packet Format

Below is a list of valid Response Codes for this request. Refer to Table 4.3 for a description of each response.

Response Code	Response Name
0x00	Success
0x02	Invalid Parameter Value
0x03	Invalid MDEP
0x04	MDEP Busy
0x05	Invalid MDL
0x06	MDL Busy
0x07	Invalid Operation
0x08	Resource Unavailable
0x09	Unspecified Error
0x0A	Request Not Supported
0x0B	Configuration Rejected

Table 4.6: Valid Response Codes for MD_CREATE_MDL_RSP

- Following a successful create operation, no other Standard Op Codes (see Standard Op Codes in Table 4.2) **shall** be sent on the Control Channel until either the L2CAP Data Channel is open for the associated Data Channel, or MD_ABORT_MDL_REQ is sent for the same MDL.
  - If the MDL ID already exists (for this MCL), then, prior to returning a “Success” Response Code, the old reference **shall** be considered obsolete and any associated Data Channel connections closed, and a new MDL created.
- The value 0xFFFF is not a valid MDL ID for this request and **shall not** be used.

The Configuration field and the Response Parameter field are intended to convey preferences/intentions for Data Channel L2CAP configuration. The meaning of non-zero values is defined by profiles that use MCAP.

- The Response Parameters for MD_CREATE_MDL_RSP **shall** comprise a single UINT 8 value containing the configuration response value.
- If the Configuration value is set to a non-zero value, then it **shall** be either accepted by repeating the same value in the Response Parameters of the

response or rejected by sending a response with Response Code “Configuration Rejected”.

- If the Configuration value is 0x00, then the Response Parameters **shall** either indicate no configuration preference with the value 0x00, or indicate a preference by using the appropriate non-zero value.

4.1.3.7.2 MD_RECONNECT_MDL

Figure 4.5: MD_RECONNECT_MDL_REQ Packet Format

Below is a list of valid Response Codes for this request. Refer to Table 4.3 for a description of each response.

Response Code	Response Name
0x00	Success
0x02	Invalid Parameter Value
0x04	MDEP Busy
0x05	Invalid MDL
0x06	MDL Busy
0x07	Invalid Operation
0x08	Resource Unavailable
0x09	Unspecified Error
0x0A	Request Not Supported

Table 4.7: Valid Response Codes for MD_RECONNECT_MDL_RSP

- Following a successful reconnect operation, no other Standard Op Codes (see Standard Op Codes in Table 4.2) **shall** be sent on the Control Channel until either the L2CAP Data Channel is open for the associated Data Channel, or MD_ABORT_MDL_REQ is sent for the same MDL.
- The value 0xFFFF is not a valid MDL ID for this request and **shall not** be used.
- The Response Parameters for MD_RECONNECT_MDL_RSP **shall** have length zero.

Because the *Source* and *Sink* assign a unique MDL ID when a new MDL is formed, using this same ID later allows a reconnection to the same context after they are intentionally closed or unintentionally broken (possibly from interference or temporarily moving out of range).

- Following an unintentional disconnect (link loss due to supervision timeout) the acceptor of the disconnected Control Channel **should** wait a reasonable amount of time[2] to allow the other device to initiate a new Control Channel before allowing its data-layer the opportunity to initiate a new Control Channel to the disconnected partner.

The normal response to receiving a rejection to a reconnect attempt is to create a new MDL ID as if interacting with a new device.

- Additional requirements related to reestablishment of Data Channel L2CAP configuration **may** be defined by profiles that use MCAP.

4.1.3.7.3 MD_ABORT_MDL

The connection of an MDL is a two-step operation (the Create or Reconnect operation, followed by the actual opening of the Data Channel), and the Abort operation is provided for the special case when this two-step operation should be cancelled after the first half has completed successfully.

- If, after issuing a MD_CREATE_MDL_REQ or MD_RECONNECT_MDL_REQ operation, a device encounters a situation that makes it no longer advisable to proceed by actually opening a Data Channel to be associated with that request, that device **may** send an MD_ABORT_MDL_REQ for the same MDL ID used in the request that initiated the transition to the PENDING state.
- When an MCL in the PENDING state receives an MD_ABORT_MDL_REQ that references the MDL ID for which it is awaiting a Data Channel connection, that MCL **shall** return to the CONNECTED or ACTIVE state, as indicated in the MCL State Machine in Figure 4.1.
- The value 0xFFFF is not a valid MDL ID for this request and **shall not** be used.

Figure 4.6: MD_ABORT_MDL_REQ Packet Format

Below is a list of valid Response Codes for this request. Refer to Table 4.3 for a description of each response.

Response Code	Response Name
0x00	Success
0x02	Invalid Parameter Value
0x05	Invalid MDL
0x07	Invalid Operation

[2] A normal choice for this delay is 12 to 15 seconds, which allows for a connecting device to make at least two connection attempts with the default page timeout of 5.12 seconds.

0x09	Unspecified Error
0x0A	Request Not Supported

Table 4.8: Valid Response Codes for MD_ABORT_MDL_RSP

- The Response Parameters for an MD_ABORT_MDL_RSP **shall** have length zero.

4.1.3.7.4 MD_DELETE_MDL

Figure 4.7: MD_DELETE_MDL_REQ Packet Format

Below is a list of valid Response Codes for this request. Refer to Table 4.3 for a description of each response.

Response Code	Response Name
0x00	Success
0x02	Invalid Parameter Value
0x05	Invalid MDL
0x06	MDL Busy
0x07	Invalid Operation
0x09	Unspecified Error
0x0A	Request Not Supported

Table 4.9: Valid Response Codes for MD_DELETE_MDL_RSP

- The reserved MDL value 0xFFFF as a parameter to this request **shall** be interpreted to indicate a request to delete all MDLs in an MCL for the requesting Bluetooth device.
- MD_DELETE_MDL **may** be initiated from either side whenever a Control Channel exists.
- If an MDL is deleted while its Data Channel is connected, the Data Channel **shall** be closed before the success Response Code is sent.
- The Response Parameters for an MD_DELETE_MDL_RSP **shall** have length zero.

4.1.3.8 Opening Data Channels

Because the channels are opened separately to a common PSM, some behavior is required to ensure a clear association of each Data Channel with the correct MDL. For example, if a device were permitted to send two MD_CREATE_MDL_REQ in succession, and then open the two associated Data Channels, variability in the physical link may cause confusion in the association of each channel to the corresponding MD_CREATE_MDL_REQ. The following steps are required to ensure stable recovery from this condition.

- When a device sends a valid MD_CREATE_MDL_REQ or MD_RECONNECT_MDL_REQ to a receiving device:

  – The receiving device **shall** move to the PENDING state to await a Data Channel connection.

  – The next L2CAP connection from the sending device to the Data Channel PSM on the receiving device **shall** be associated with the MDL referenced in the MD_CREATE_MDL_REQ or MD_RECONNECT_MDL_REQ that caused the receiving device to enter the PENDING state.

- While a device is in the PENDING state, it **shall** send an “Invalid Operation” response to any Standard Op Codes other than a valid MD_ABORT_MDL, but **shall** still process Clock Synchronization Protocol Op Codes as it normally would.

4.2 Data Channel Protocol

Each open MDL comprises one Data Channel, which is created following an MD_CREATE_MDL or MD_RECONNECT_MDL operation on the Control Channel of the MCL (see description of PENDING state in Table 4.1 and Opening Data Channels in Section 4.1.3.8). A Data Channel carries the application content as described in the Data Exchange Specifications which define a base data protocol and data formats.

The actual traffic on the Data Channel connection is completely defined by the Data Exchange Specifications. A profile making use of MCAP defines the specific Data Exchange Specifications to be used.

- If a Data Channel connection attempt (an L2CAP connection on the Data Channel PSM) is received from a device that has not sent a successful Create or Reconnect operation (i.e. the state machine of the MCL with that remote device is not in the PENDING state), that connection **should** be refused at the L2CAP layer using L2CAP Connection Response “Connection refused – no resources available”.

  – When it is not feasible to refuse the connection directly from L2CAP, such a connection **shall** be closed as quickly as possible, without sending (or responding to) any data traffic.

4.3 Support for Protocol Requirements

The Clock Synchronization Protocol (see Section 5) is normally used in combination with Data Channels to provide a service, although this will not always be the case. It is normally the case that Standard Op Codes will be supported by an MCAP implementation, but support for the Clock Synchronization Protocol might be used in combination with some other data transport that is not compatible with MCAP Data Channels.

- Implementations of MCAP **should** support all Standard Op Codes.

  – When Standard Op Codes are not supported, all Clock Synchronization Protocol Op Codes **shall** be supported.

The table below applies only to implementations using Standard Op Codes and shows the required support for MCAP features for *Sources* and *Sinks.*

- If an implementation claims support for Standard Op Codes, it **shall** comply with the requirements shown in Table 4.10. Otherwise, the features below are Not Applicable.

MCAP Feature	**Source Status**	**Sink Status**
Initiate creation of Control and Data Channels	C.1	M
Accept creation of Control and Data Channels	C.1	M
Initiate Disconnection of MCL	M	M
Accept Disconnection of MCL	M	M
Initiate Disconnection of MDL	M	M
Accept Disconnection of MDL	M	M
Initiate Reconnection of MDL	O	O
Accept Reconnection of MDL	C.2	M
Initiate Deletion of MDL	O	O
Accept Deletion of MDL	M	M
Initiate Delete of All MDLs using 0xFFFF	O	O
Accept Delete of All MDLs using 0xFFFF	M	M
Send MDL Abort request	O	O
Accept MDL Abort request	M	M

Table 4.10: Support for MCAP Features

C.1: Support for at least the initiate or accept feature is Mandatory.

C.2: Mandatory if Accept creation of Control and Data Channels is supported, otherwise Excluded.

5 Clock Synchronization Protocol

The Clock Synchronization Protocol (CSP) is optional.

- If a device does not support the Clock Synchronization Protocol, but receives a Clock Synchronization Protocol Op Code, that device **shall** respond with a "Request Not Supported" Response Code and the Op Code field of the Response Packet **shall** be set to the Response Op Code that corresponds to the unsupported Request Op Code.

  – If a device sends a Clock Synchronization Protocol Op Code and receives a "Request Not Supported" response, the device **shall not** send any further clock synchronization request or indication operations for the duration of that Control Channel instance.

- If a device does not support the Clock Synchronization Protocol, that device **shall not** send any packets that use Clock Synchronization Protocol Op Codes.

5.1 Clock Synchronization Overview

Precise timing synchronization between the input or output of two or more devices in the millisecond or even microsecond range is necessary for various applications, including synchronized data collection from several high speed sensors, high accuracy sensing of parameters such as vibration, acceleration, propagation delay measurements, or coordinated event initiation. All these applications need not only the information *that* a specific event occurred (or will occur), but also the precise declaration of *when* it occurred (or will occur) in order to combine (or coordinate) the information of multiple devices.

To enable the aforementioned types of applications, a highly precise timing synchronization of the data from Bluetooth-connected devices is needed. Since Bluetooth devices use time-slotted radio links with error correction based on lower layer retransmission, there is no fixed end-to-end propagation delay provided that could be used to grant a high timing accuracy out of the transported data itself. So, to allow the precise timed reassembly or coordination of the data of multiple devices, a time-stamp mechanism for the data of each data *Source* is required.

The protocol that is described in this section can be used to precisely synchronize the time-stamps of one or more Sync-Slaves that are connected to a Sync-Master via a Bluetooth radio link or allow the recalculation and timing error correction of the timing of events. The achieved precision with this protocol is dependent on the precision with which implementations can access the Bluetooth Clock. This protocol is designed for use between devices which are on the same piconet.

One hypothetical example for a use case that uses both, the ability to lock the time-stamp to an "absolute time" and the ability to put event timing from multiple events from one or more *Sources* in "relative time" relative to each other, is shown in Figure 5.1 below:

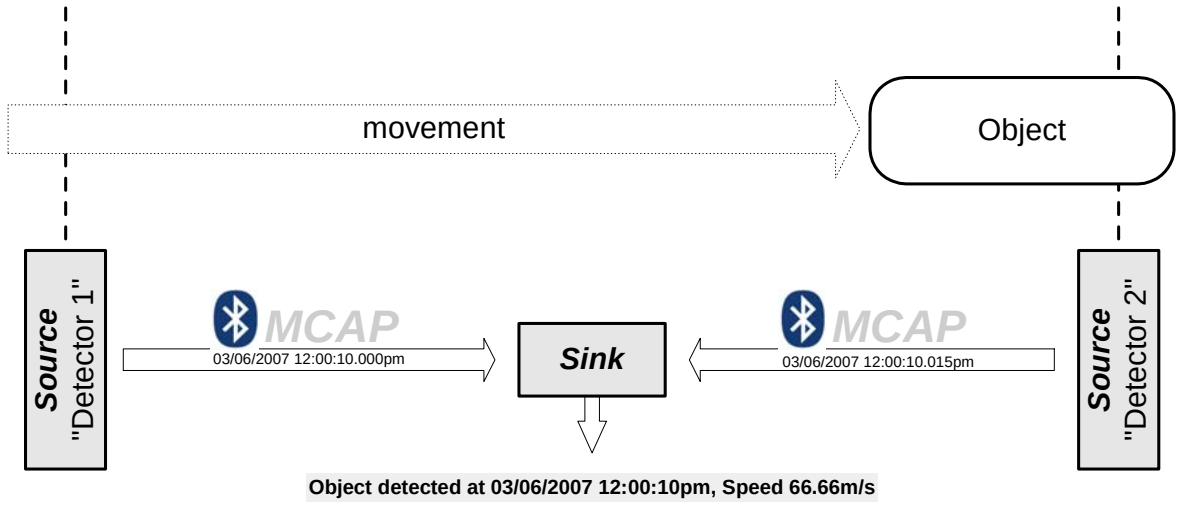

Figure 5.1: Time Synchronization Example

Two *Sources*, each with an optical detector, send the "interrupted/not interrupted" status from these detectors via MCAP connections to a *Sink* that calculates the direction and speed of an object moving through these detectors.

The task of the *Sink* in this example is to generate an event list of the kind "at 03/06/2007 12:00:10pm an object with the speed of 66.66m/s was detected"

To fulfill the task, the *Sink* configures the time-stamps of the *Sources* in a way that they represent "absolute time" instances. As an example, this could be done by converting an IEEE 1003.1-2001 [4] conformant time definition (this standard basically counts the seconds since January 01, 1970) to the time-stamp's microsecond resolution. Once that is done, the highly precise timing of each event marked with an MCAP time-stamp can easily be reconverted to a fully conformant IEEE 1003.1-2001 [4] "absolute time", and the "relative timing" of the both events can easily be determined to calculate the object's speed.

Refer to Section 6.2.3 for examples of Clock Synchronization Transactions.

5.1.1 Bluetooth Clock

Every Bluetooth device has an internal system clock which determines the timing and hopping of the transceiver. If two devices have a Bluetooth connection, a common Bluetooth clock negotiation is performed as defined in the Bluetooth Core Specification [1].

It should be noted that the Bluetooth Clock has no relation to the time of day. The Bluetooth Clock provides the "heart beat" of the Bluetooth transceiver. Its resolution is half the TX or RX slot length, or 312.5 µs.

With the requests and responses described in Section 5.2, the Bluetooth Clock, which is common to all devices in an a piconet, is used to synchronize a protocol implementation internal Time-Stamp Clock and respectively allows the recalculation of

the precise Time-Stamp timing if at least the Sync-Slave is capable of accessing its Bluetooth Clock.

- The radio link Bluetooth Clock that is used as a time base for synchronization **can** be accessed via the HCI_READ_CLOCK command that is defined in the Bluetooth Core Specification [1].

Internal hardware, threading, processing, and polling delays in accessing the Bluetooth Clock are incorporated in the SyncLeadTime field defined below, and are important considerations when determining accuracy and required update frequency.

5.1.1.1 Baseband Half-Slot Instant

A Baseband Half-Slot Instant consists of a half-slot counter value at which an event happens.

5.1.2 Time-Stamp Clock

The Time-Stamp Clock is a clock with a resolution of one microsecond (μs). It is defined as a 64 bit counter.

- The Time-Stamp Clock **can** be initialized by a Sync-Master to an arbitrary value at any time.
- The Time-Stamp Clock **can** be used as a synchronous running time source for communication in an MCL.
- The Time-Stamp Clock **shall** be normalized to microseconds (μs).

5.1.2.1 Time-Stamp Instant

A Time-Stamp Instant consists of a microsecond counter value at which an event happens.

5.1.3 Clock Synchronization Roles

The Clock Synchronization Protocol defines the roles of Sync-Master and Sync-Slave.

5.1.3.1 Sync-Master

The device that requests time-stamp information, or configures synchronization requirements on a remote device is called Sync-Master for the time the corresponding request is processed by the Sync-Slave.

5.1.3.2 Sync-Slave

The device that receives a request for time-stamp information, or receives configuration synchronization requirements from a remote device master is called Sync-Slave for the time the corresponding request is processed by the Sync-Slave.

5.1.3.3 Role Requirements

This section describes requirements for the support of these roles.

- Within a single MCL instance at any particular time, an implementation of the Clock Synchronization Protocol **shall** support Sync-Slave functionality and **may** support Sync-Master functionality.
- A device **can** be Sync-Master and Sync-Slave at the same time.

Note that although the Sync-Slave functionality forms the minimal core of the CSP, some reserved values have been created to allow Sync-Master devices that do not have access to their Bluetooth Clocks to respond to Sync-Slave operations (i.e. support the Sync-Slave functionality). These include the BluetoothClock_AccessResolution reserved value 0x00 (see Section 5.2.3.1) and the SYNC_SET_RSP response code for "Resource Unavailable" (see Section 5.2.3.2).

5.2 Clock Synchronization Requests and Responses

5.2.1 Clock Synchronization Protocol Response Packet Format

The Clock Synchronization Protocol Response Packet format applies to all *_RSP codes defined in Table 5.1 (i.e., MD_SYNC_CAP_RSP and MD_SYNC_SET_RSP).

Op Code (Clock Sync)	Response Code	Response Parameters
UINT 8	UINT 8	Variable

Figure 5.2: Response Packet Format (Clock Synchronization Protocol Op Code)

5.2.2 Clock Synchronization Protocol Op Codes

Clock Sync Op Code	Request / Response Name	Description
0x11	MD_SYNC_CAP_REQ	Request synchronization capabilities and requirements
0x12	MD_SYNC_CAP_RSP	Indicate completion of synchronization capabilities operation
0x13	MD_SYNC_SET_REQ	Request to set Time-Stamp Clock at a time specified in terms of the Bluetooth Clock
0x14	MD_SYNC_SET_RSP	Indicate that the requested SET has been completed
0x15	MD_SYNC_INFO_IND	Includes an update of the actual Time-Stamp Clock Instant from the Sync-Slave
0x16	Reserved[3]	There is no response for the MD_SYNC_INFO_IND indication
0x17-0x20	Reserved	N/A

Table 5.1: Clock Synchronization Protocol Request / Response Summary

The Request Packet format is shown in Figure 4.2 and the Response Packet format is shown in Figure 5.2.

5.2.3 Requests and Responses

5.2.3.1 MD_SYNC_CAP

With this REQ/RSP pair, the Sync-Master informs the Sync-Slave about its synchronization requirements.

[3] To preserve generality, the Op Code MD_SYNC_INFO_IND+1 (which would normally be used as the Op Code of the associated response) has been reserved.

The packet structure for MD_SYNC_CAP_REQ is shown in Table 5.2.

Byte	Field Name	Unit	Type
1	MD_SYNC_CAP_REQ	N/A	UINT 8
2-3	TimeStamp_RequiredAccuracy	ppm	UINT 16

Table 5.2: MD_SYNC_CAP_REQ Packet Structure

The fields for MD_SYNC_CAP_REQ are described below.

TimeStamp_RequiredAccuracy

This field defines the accuracy requirements of the Sync-Master (the initiator of this request) for the common Time-Stamp Clock.

The packet structure for MD_SYNC_CAP_RSP is shown in Table 5.3.

Byte	Field Name	Unit	Type
1	MD_SYNC_CAP_RSP	N/A	UINT 8
2	Response Code	N/A	UINT 8
3	BluetoothClock_AccessResolution	Baseband half-slots	UINT 8
4-5	SyncLeadTime	ms	UINT 16
6-7	TimeStamp_NativeResolution	μs	UINT 16
8-9	TimeStamp_NativeAccuracy	ppm	UINT 16

Table 5.3: MD_SYNC_CAP_RSP Packet Structure

The fields for MD_SYNC_CAP_RSP are described below.

BluetoothClock_AccessResolution

If the responding protocol implementation is capable of accessing the Bluetooth Clock, this value defines the resolution of the Bluetooth Clock that can be accessed by that implementation. Otherwise it has to be set to zero (0x00).

SyncLeadTime

This field defines the minimum time offset (from the receipt of a MD_SYNC_REQ) that is required by the responding protocol implementation for internal processing. Internal hardware, threading, processing, and polling delays in accessing the Bluetooth Clock are incorporated in the SyncLeadTime.

TimeStamp_NativeResolution

This field defines the native time resolution of the Time-Stamp Clock implementation.

TimeStamp_NativeAccuracy

This field defines the worst case accuracy of the Time-Stamp Clock of the device.

Response Code descriptions for MD_SYNC_CAP_RSP are shown in Table 5.4.

Response Code	Response Name	Condition
0x00	Success	The corresponding request was received and processed successfully.
0x02	Invalid Parameter Value	The Parameter value or length was inappropriate for the command
0x09	Unspecified Error	An internal error other than those listed in this table was encountered while processing the request
0x0A	Request Not Supported	The Op Code that was used in this request is not supported

Table 5.4: MD_SYNC_CAP_RSP Response Code Description

- If the Response Code indicates a response other than Success (0x00), the Response Parameters (bytes 3-9) **shall** be set to zero.

If two or more devices have a Bluetooth connection, a common timing scheme is available because the slot timing in each connection is based on the link master's Bluetooth Clock. The accuracy of this clock is dependent upon the operating mode(s) of the link master (active versus low power modes) and the accuracy of the internal oscillator(s) used by the piconet master.

- If an implementation is allowed to enter Sniff mode during a MD_SYNC_SET operation, this **should** be considered for the worst case for the TimeStamp_NativeAccuracy value.

5.2.3.2 MD_SYNC_SET

With this REQ/RSP pair, the Sync-Master initiates a Synchronization, or Re-Synchronization process of the Sync-Slave Time-Stamp Clock to the Sync-Master Time-Stamp Clock.

The packet structure for MD_SYNC_SET_REQ is shown in Table 5.5.

Byte	Field Name	Unit	Type
1	MD_SYNC_SET_REQ	N/A	UINT 8
2	TimeStamp_UpdateInformation	Boolean	UINT 8
3-6	BluetoothClock_SyncTime	Baseband Half-Slot Instant	UINT 32
7-14	TimeStamp_SyncTime	Time-Stamp Clock Instant	UINT 64

Table 5.5: MD_SYNC_SET_REQ Packet Structure

The fields for MD_SYNC_SET_REQ are described below.

TimeStamp_UpdateInformation

If set to 1, this field indicates the request for sending the MD_SYNC_INFO_IND indicator to the Sync-Master (the initiator of this request). The only other valid value is 0.

BluetoothClock_SyncTime

Bluetooth Clock time half slot at which synchronization is requested, or 0xFFFFFFFF to request instant synchronization

TimeStamp_SyncTime

This field indicates the Time-Stamp Clock position to set at the requested Bluetooth Clock time. Otherwise 0xFFFFFFFF FFFFFFFF is used to indicate that no set should take place (used to receive the response information only).

The packet structure for MD_SYNC_SET_RSP is shown in Table 5.6.

Byte	Field Name	Unit	Type
1	MD_SYNC_SET_RSP	N/A	UINT 8
2	Response Code	N/A	UINT 8
3-6	BluetoothClock_SyncTime	Baseband Half-Slot Instant	UINT 32
7-14	TimeStamp_SyncTime	Time-Stamp Clock Instant	UINT 64
15-16	TimeStamp_SampleAccuracy	μs	UINT 16

Table 5.6: MD_SYNC_SET_RSP Packet Structure

The fields for MD_SYNC_SET_RSP are described below.

BluetoothClock_SyncTime

This field defines the Bluetooth Clock at the send time of this response.

TimeStamp_SyncTime

This field defines the Time-Stamp Clock at the send time of this response.

TimeStamp_SampleAccuracy

This field defines the Maximum error for this clock sample, including the combined maximum errors in triggering the initial clock set and in writing the set value, as well as the maximum absolute difference between the actual read times of the two clock values included in this indication packet.

Response Code descriptions for MD_SYNC_SET_RSP are shown in Table *5.7*.

Response Code	Response Name	Condition
0x00	Success	The corresponding request was received and processed successfully.
0x02	Invalid Parameter Value	The Parameter value or length was inappropriate for the command including: - TimeStamp_UpdateInformation was set to 1, but no MD_SYNC_CAP_REQ was received, so no accuracy requirements are known. - TimeStamp_UpdateInformation was set to a value other than 0 or 1 - BluetoothClock_SyncTime was set to an illegal Bluetooth Clock

		time slot number other than 0xFFFFFFFF - The Baseband Half-Slot Instant indicated by BluetoothClock_SyncTime was less than SyncLeadTime in the future at the time the request was received - The calculated Time-Stamp Update Interval for the MD_SYNC_INFO_IND is less than the "SyncLeadTime" - The Baseband Half-Slot Instant indicated by BluetoothClock_SyncTime was more than SyncLeadTime plus 60 seconds in the future at the time the request was received - BluetoothClock_SyncTime was set to a legal Bluetooth Clock time slot number, but no MD_SYNC_CAP_REQ was received.
0x07	Invalid Operation	The received request is invalid in the current state. - A piconet role-switch was performed after receipt of the MD_SYNC_SET_REQ but before Time-Stamp synchronization
0x08	Resource Unavailable	BluetoothClock_SyncTime was set, but protocol implementation has no access to the Bluetooth Clock, so synchronization can not be performed
0x09	Unspecified Error	An internal error other than those listed in this table was encountered while processing the request.
0x0A	Request Not Supported	The Op Code that was used in this request is not supported

Table 5.7: MD_SYNC_SET_RSP Response Code Description

- If the Response Code indicates a response other than Success (0x00), the Response Parameters (bytes 3-16) **shall** be set to zero.

Further requirements for MD_SYNC_SET are described below.

- If the "BluetoothClock_SyncTime" is set to 0xFFFFFFFF (illegal Bluetooth Clock time slot number, see Bluetooth Core Specification [1] for details), the Sync-Slave **shall** immediately set the Time-Stamp Clock to the "TimeStamp_SyncTime" indicated value, and then **shall** respond with a MD_SYNC_SET_RSP.
- If the "BluetoothClock_SyncTime" is set to another valid value (see MD_SYNC_SET_RSP description for details), the Sync-Slave **shall** set the Time-Stamp Clock to the "TimeStamp_SyncTime" indicated value when the Bluetooth Clock time reaches the Baseband Half-Slot Instant defined in the "BluetoothClock_SyncTime" and then **shall** respond with a MD_SYNC_SET_RSP.
- If "TimeStamp_UpdateInformation" is set to 1, the Sync-Slave **shall** start to periodically send MD_SYNC_INFO_IND indicators after sending the MD_SYNC_SET_RSP within the "SyncLeadTime" time interval (for more information see MD_SYNC_INFO_IND description)
  - If the calculated Time-Stamp Update Interval is less than the "SyncLeadTime" the Sync-Slave **shall** immediately respond with a MD_SYNC_SET_RSP with response code "Invalid Parameter Value"
- If the TimeStamp_SyncTime is set to 0xFFFFFFFF FFFFFFFF, then the value of the TimeStamp clock **shall not** be reset at the designated moment.

MD_SYNC_SET_RSP (and MD_SYNC_INFO_IND, if appropriate) **shall** be sent as normal, including the actual TimeStamp clock value.

5.2.3.3 MD_SYNC_INFO_IND

This indicator includes a "time tuple" of Time-Stamp- and Bluetooth Clock and basically indicates to the Sync-Master “At this instant in time, my Bluetooth Clock was at count X, and my Time-Stamp Clock was at count Y. The maximum error included in this calculation is Z”.

The send period of this MD_SYNC_INFO_IND indicator is called the Time-Stamp Update Interval and is calculated as the TimeStamp_RequiredAccuracy (provided by the Sync-Master as part of the MD_SYNC_CAP_REQ command) divided by the TimeStamp_NativeAccuracy (provided by the Sync-Slave as part of the MD_SYNC_CAP_RSP response).

For example, if the Sync-Master indicates a TimeStamp_RequiredAccuracy of 1000ppm, and the Sync-Slave indicates a worst case TimeStamp_NativeAccuracy of 20ppm, the Sync-Slave is required to send a MD_SYNC_INFO_IND to the Sync-Master at least every 1000/20 or 50 seconds.

The packet structure for MD_SYNC_INFO_IND is shown in Table 5.8.

Byte	Field Name	Unit	Type
1	MD_SYNC_INFO_IND	N/A	UINT 8
2-5	BluetoothClock_SyncTime	Baseband Half-Slot Instant	UINT 32
6-13	TimeStamp_SyncTime	Time-Stamp Clock Instant	UINT 64
14-15	TimeStamp_SampleAccuracy	µs	UINT 16

Table 5.8: MD_SYNC_INFO_IND Packet Structure

The fields for MD_SYNC_INFO_IND are described below.

BluetoothClock_SyncTime

This field defines the Bluetooth Clock at the send time of this indicator.

TimeStamp_SyncTime

This field defines the Time-Stamp Clock at the send time of this indicator.

TimeStamp_SampleAccuracy

This field defines the Maximum error for this clock sample, including the combined maximum errors in triggering the initial clock set and in writing the set value, as well as the maximum absolute difference between the actual read times of the two clock values included in this indication packet.

Further requirements for MD_SYNC_INFO_IND are described below.

- If the Sync-Slave allows low power modes while MD_SYNC_INFO_IND messages are requested, the Sync-Slave **shall** ensure that the negotiated power-down

intervals (like sniff intervals) do not exceed the send period of the MD_SYNC_INFO_IND messages.

- This indicator **shall** be periodically sent by a Sync-Slave if the last received MD_SYNC_SET_REQ includes a TimeStamp_UpdateInformation flag set to 1
- The first MD_SYNC_INFO_IND message **shall** be sent by the Sync-Slave within the SyncLeadTime indicated by the last MD_SYNC_CAP_RSP.
- Once a Sync-Slave has started sending MD_SYNC_INFO_IND indicators, the Sync-Slave **shall** continue to periodically send these indicators as long as the MCAP connection is established, or a MD_SYNC_SET_REQ with TimeStamp_UpdateInformation set to 0 is received.
- A Sync-Slave **can** continue to periodically send MD_SYNC_INFO_IND after reestablishment of an MCAP connection.
  - In this case, the Sync-Slave **shall** send the first MD_SYNC_INFO_IND after reestablishment of the MCAP connection to the Sync-Master within the SyncLeadTime indicated by the last MD_SYNC_CAP_RSP.

There is no response to this indicator. To preserve generality, the Op Code MD_SYNC_INFO_IND+1 (which would normally be used as the Op Code of the associated response) has been reserved.

5.3 Using Clock Synchronization Protocol with Multiple MCLs

When performing clock synchronization using multiple MCLs, different MCLs may require different time stamps. For this condition, the following requirements apply.

- Clock Synchronization Protocol operations **shall** apply only to the MCL on which the operations are carried.
- Clock Synchronization Protocol operations **shall** affect timer values only on the MCL associated with the Control Channel on which the operations are carried.

5.4 Support for Protocol Requirements

The table below shows the required support for Clock Synchronization Protocol operations for devices supporting the Sync-Slave and Sync-Master roles.

- If an implementation claims support for the Clock Synchronization Protocol, it **shall** comply with the requirements shown in Table 5.9. Otherwise, the features below are Not Applicable.

MCAP CSP Feature	Sync-Slave	Sync-Master
Initiate MD_SYNC_CAP_REQ and MD_SYNC_SET_REQ and Accept MD_SYNC_INFO_IND	X	M
Accept MD_SYNC_CAP_REQ and MD_SYNC_SET_REQ and Initiate MD_SYNC_INFO_IND	M	X

Table 5.9: Support for CSP Features

6 MCAP Transaction Examples

6.1 Standard Op Code Examples

The following examples are presented as a sequence.

The example in Section 6.1.1 shows a basic case where a device such as a set top box (*Sink*) is discovered by a *Source*. The two devices connect, their data layers discover each others capabilities (saving those for later use) and disconnect.

In the example in Section 6.1.2, the *Source* has a measurement ready for transmission, so it reconnects to the *Sink*, sends the measurement and the *Source* disconnects.

In the example in Section 6.1.3, the *Sink* requests a measurement from the *Source*. The *Sink* reconnects to the *Source*, the *Source* sends the measurement, and the *Sink* disconnects.

The examples in Sections 6.1.4 and 6.1.5 show connection to a multi-function device with multiple Data Channels of different data types (e.g. a low-bandwidth channel and a high-bandwidth channel). These examples show different ways in which these Data Channels can be setup, used to exchange measurement data and closed.

The example in Section 6.1.6 shows some variations allowed for opening and closing of channels.

The examples in Sections 6.1.7 and 6.1.8 represent typical failure modes (Invalid MDL and Involuntary Disconnect).

6.1.1 Simple Case

In this case, *Source* A initiates a Control Channel connection to *Sink* B, creates an MDL, uses it to exchange data, and then disconnects by first closing the Data Channel followed by the Control Channel. Although this example shows the common case where the device which initiated the connection also closed the connection, either side is able to close the connection regardless of which side opened it.

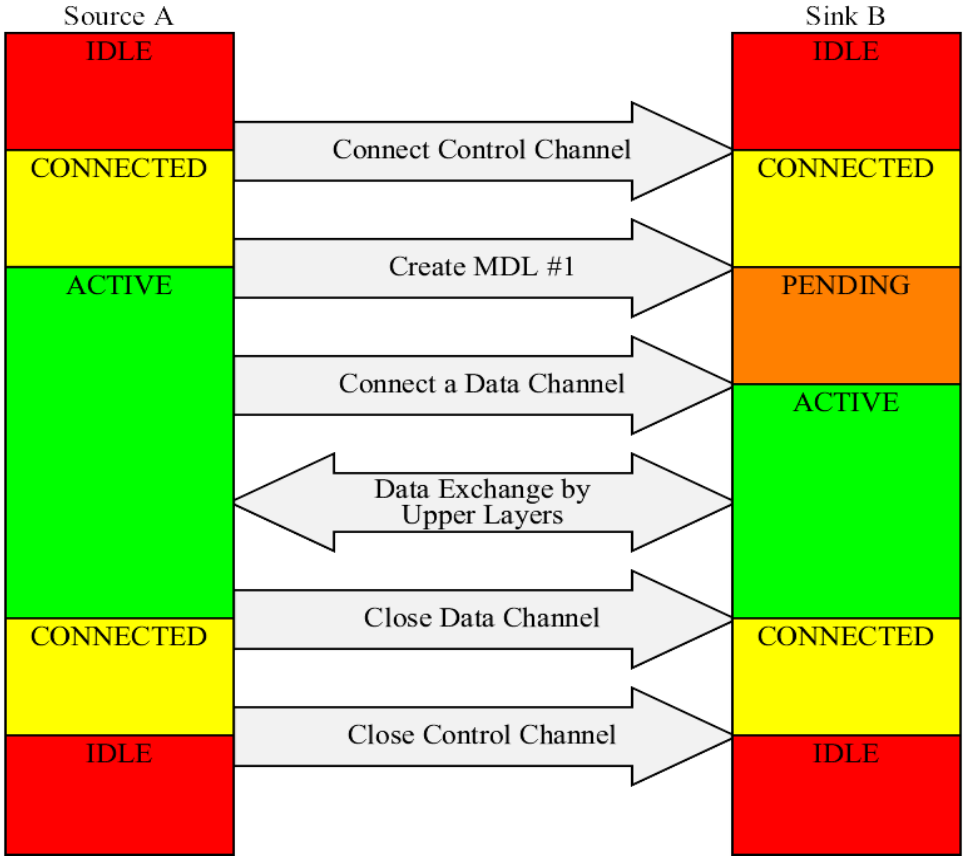

Figure 6.1: Protocol Transaction Example - Simple Case

6.1.2 Reconnect - Simple Case - Source Initiated

This case assumes the previous case has occurred and the devices have previously connected to each other, discovered each others capabilities and *Source* A had already defined MDL #1 for *Sink* B. This example shows the use of the Reconnect feature for a device making periodic measurements where the *Source* is initiating a reconnection back to the *Sink*.

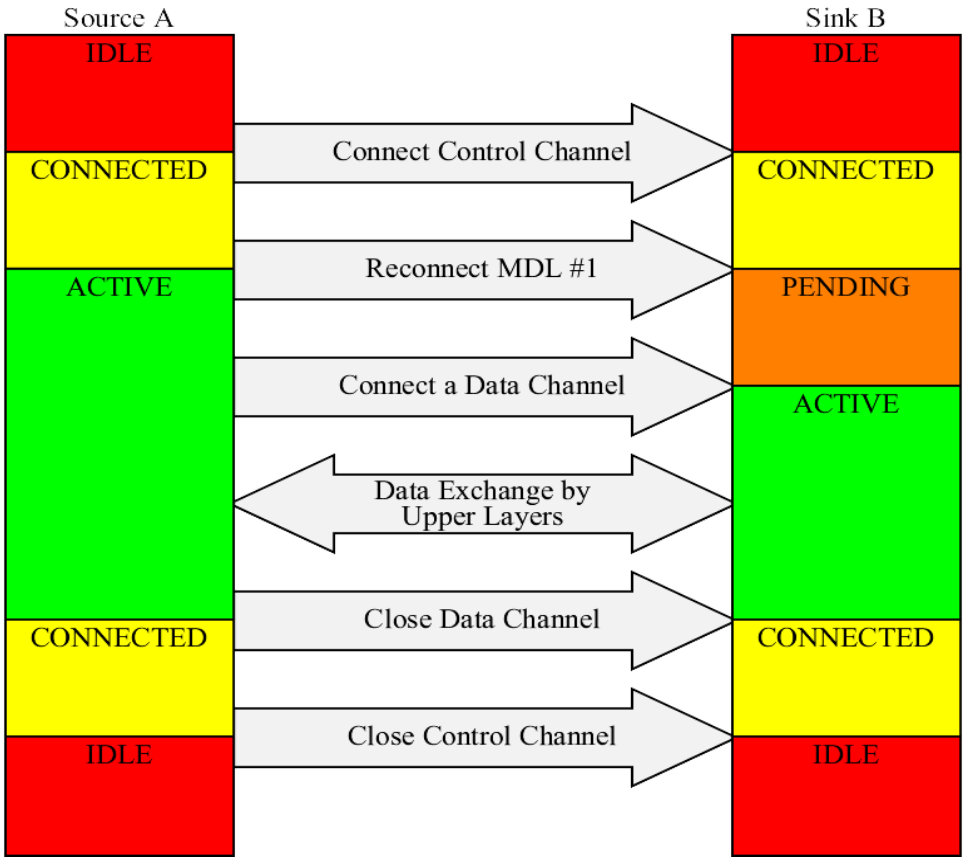

Figure 6.2: Protocol Transaction Example - Source Initiated Reconnect

The use of Reconnect allows configuration exchanges to be optimized by not having to re-exchange already-known data. Device A uses the MDL to exchange data, and then disconnects by first closing the Data Channel followed by the Control Channel.

Note that a Source-initiated request for measurement data may be done on a periodic basis assuming the data layer permits this type of request.

6.1.3 Reconnect - Simple Case - Sink Initiated

This case is the same as the previous case, except that in this example, *Sink* B initiates reconnection of the MDL that has been initially created by *Source* A.

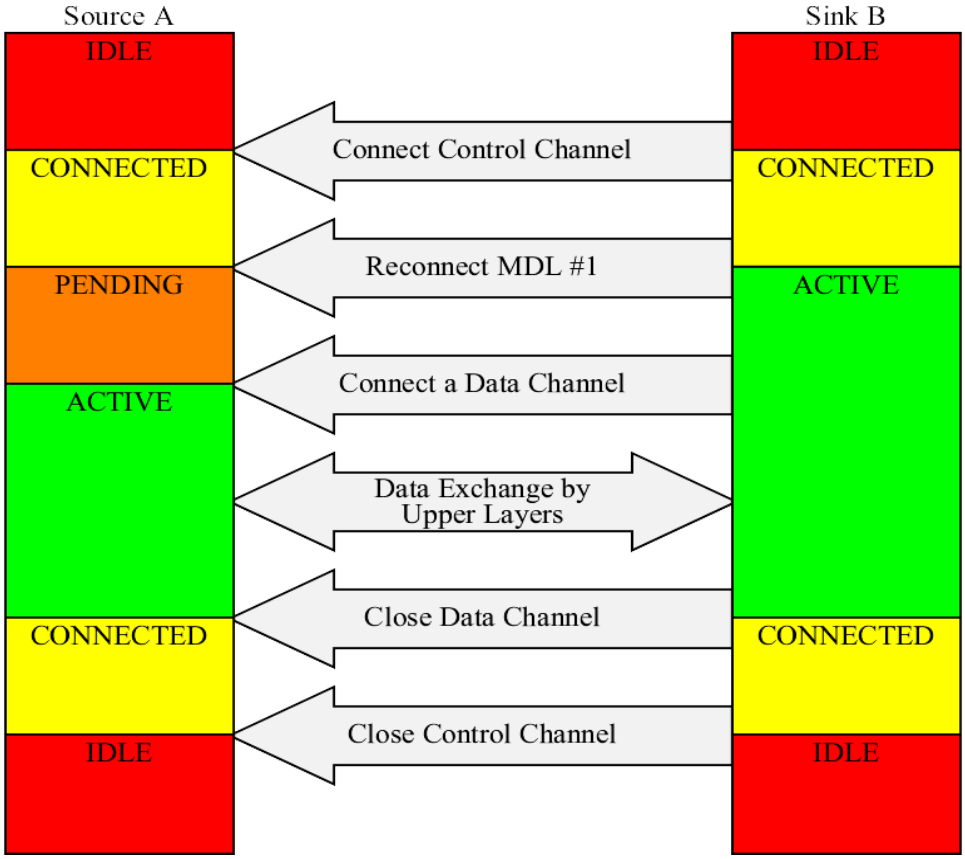

Figure 6.3: Protocol Transaction Example - Sink Initiated Reconnect

Note that this example also illustrates that either side can connect and disconnect.

Note that although MDL #1 was created by the *Source* in the case shown in 6.1.1, either the *Source* or the *Sink* are allowed to reconnect to it.

6.1.4 Multiple Data Channels - Parallel Case

In this example, *Source* A uses two Data Channels in parallel (i.e. at the same time) to communicate with a new partner (*Sink* C). *Source* A exchanges data on the first Data Channel before opening the second Data Channel. Although *Source* A had already defined MDL #1 for *Sink* B in the previous example, it is free to use MDL #1 for *Sink* C without ambiguity, since MDLs are unique only for each pair of devices.

This case also uses the feature that, by closing the Control Channel, all Data Channels are automatically closed, returning both devices directly to IDLE without an intermediate CONNECTED state.

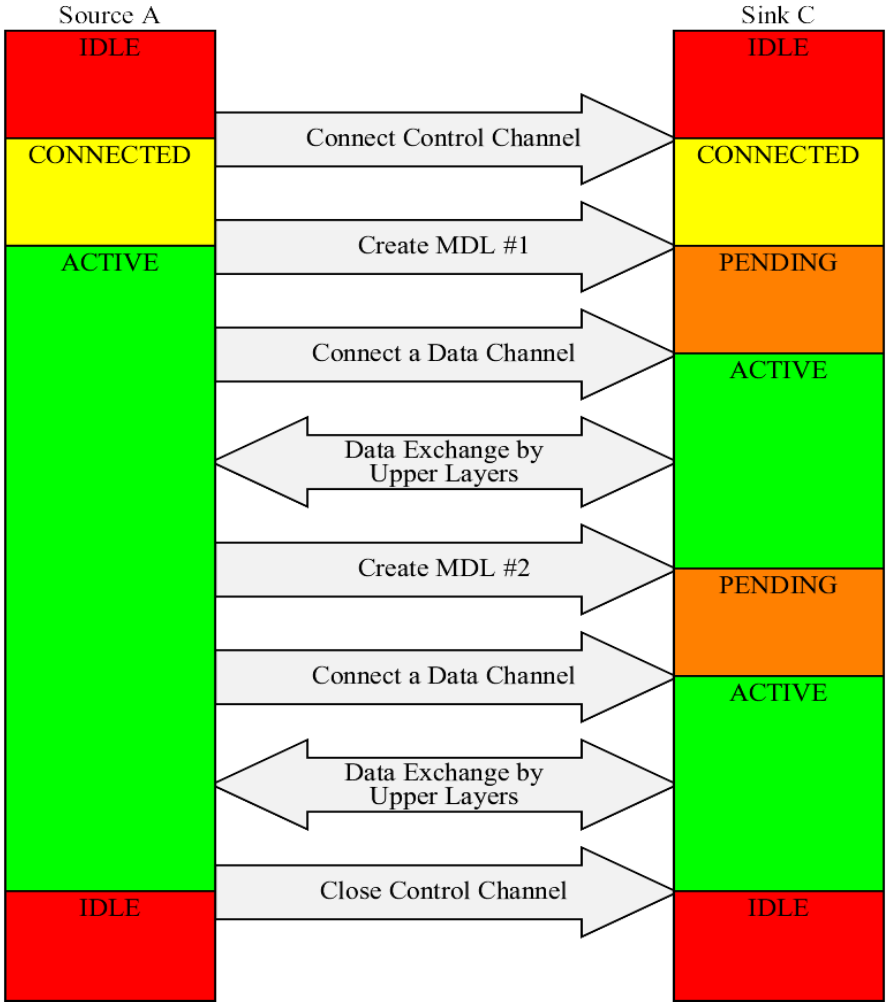

Figure 6.4: Protocol Transaction Example - Multiple Data Channels - Parallel Case

Note that in this case, the exchange of application data on the first Data Channel may continue while device C is PENDING in anticipation of the connection of the second Data Channel. No activity is valid on the Control Channel while a device is PENDING, but the existing Data Channels are not affected.

6.1.5 Multiple Data Channels - Sequential Case

Similar to previous case, *Source* A uses two Data Channels to communicate with a new partner (*Sink* C); however, these Data Channels are used sequentially rather than in parallel. This example illustrates that *Source* A can exchange data on the first Data Channel and close it before opening the second Data Channel for use with no need to reconnect to the Control Channel.

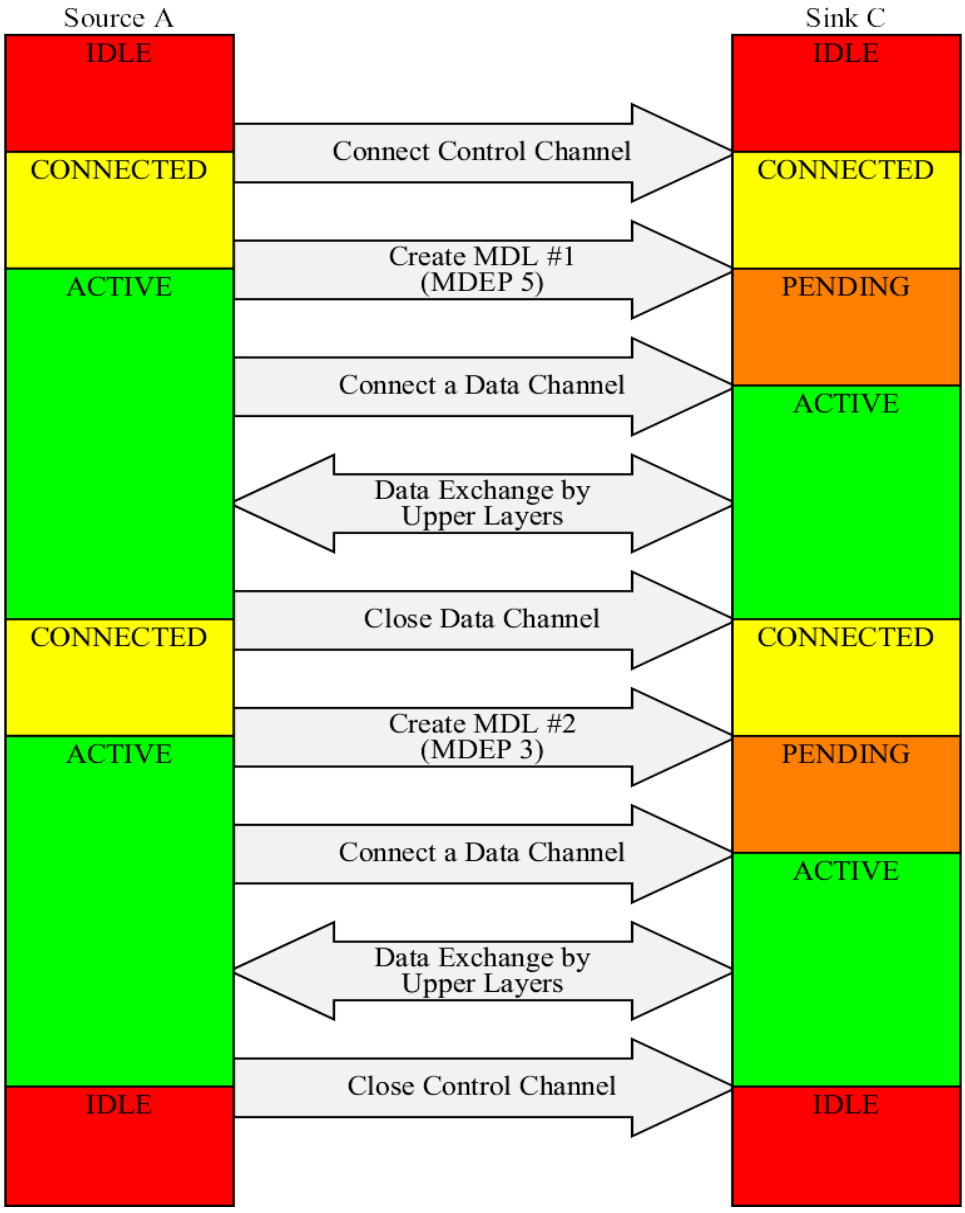

Figure 6.5: Protocol Transaction Example - Multiple Data Channels - Sequential Case

6.1.6 Opening and Closing Data and Control Channels

This example illustrates that the device that initiates a connection to a Control Channel does not necessarily need to be the device which opens Data Channels. It also illustrates that the device accepting the Control Channel connection may close the Control Channel.

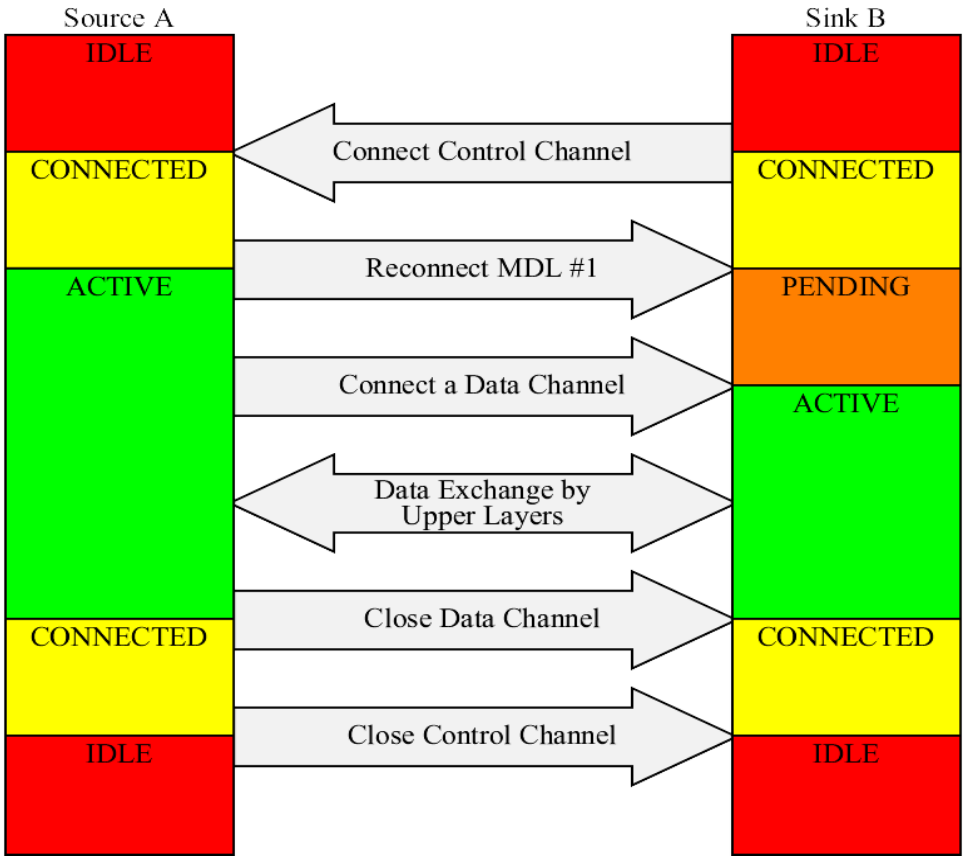

Figure 6.6: Protocol Transaction Example - Opening and Closing Data and Control Channels

6.1.7 Failure Case - Invalid MDL

In this example, *Source* A attempts to reconnect to *Sink* C using the already defined MDL #5. If for some reason, *Sink* C returns Invalid MDL, *Source* A then deletes MDL #5, closes all of its associated data connections, and creates a new data connection using MDL #5.

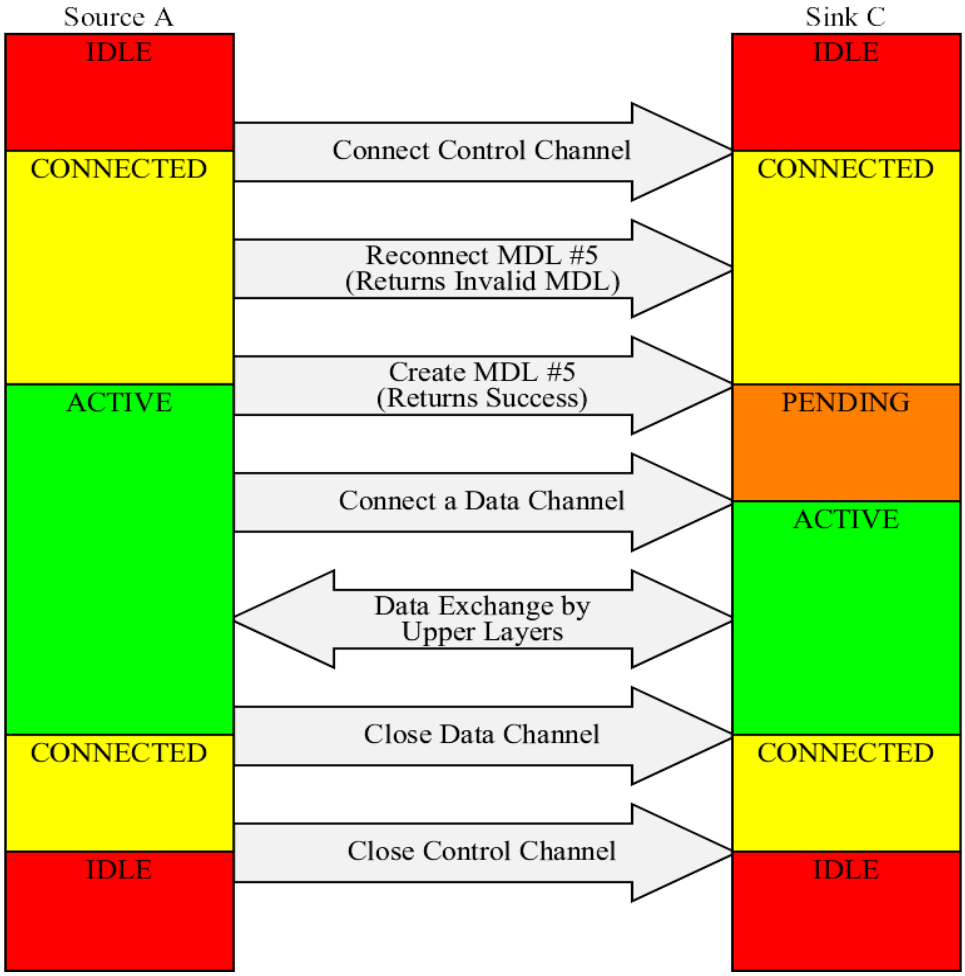

Figure 6.7: Protocol Transaction Example - Failure Case - Invalid MDL

6.1.8 Involuntary Disconnect Case

In this example, *Source* A and *Sink* B connect and begin exchanging data. If for some reason the connection is lost, both devices return to the IDLE state.

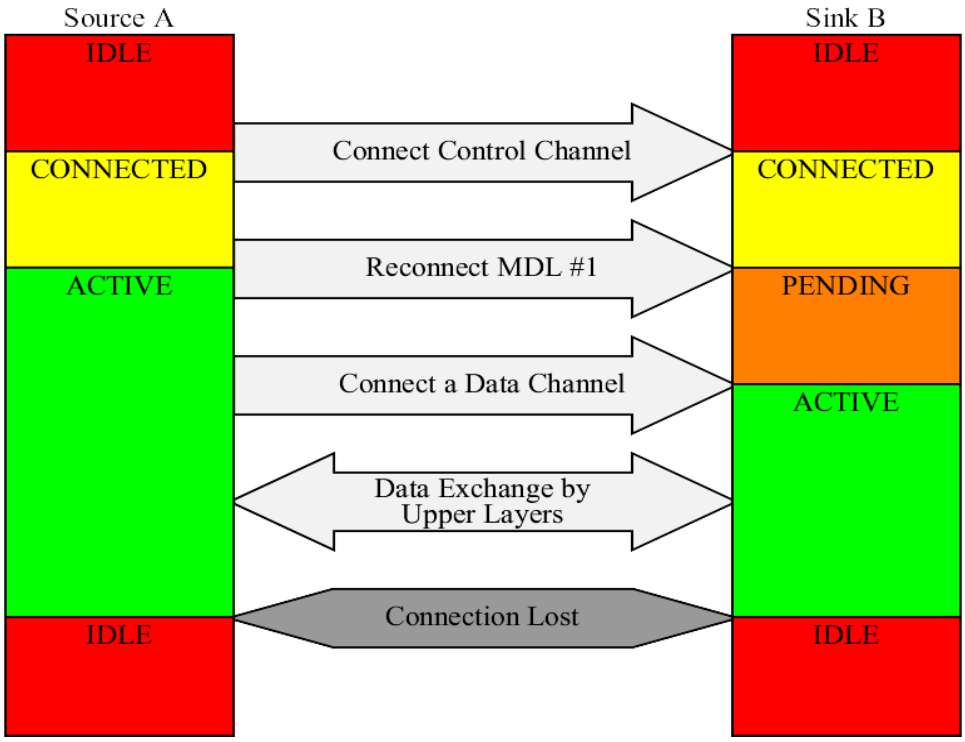

Figure 6.8: Protocol Transaction Example - Involuntary Disconnect Case

6.2 Clock Synchronization Protocol Examples

The examples in Sections 6.2.1 through 6.2.3 show examples of typical clock synchronization use including a simple case, a standard case and an update request case.

6.2.1 Clock Synchronization - Simple Case

This diagram shows a simple clock synchronization interaction, in which the Sync-Master initiates the Control Channel connection, confirms the synchronization capabilities of the Sync-Slave, then requests a clock set event for a time in the near future (when the shared Bluetooth master clock reaches value x). The connection then proceeds as usual. When the Bluetooth master clock reaches time x, the Sync-Slave sets its Time-Stamp Clock to the value requested in the "Set" request. The example shows that the CSP can work without MCAP standard commands.

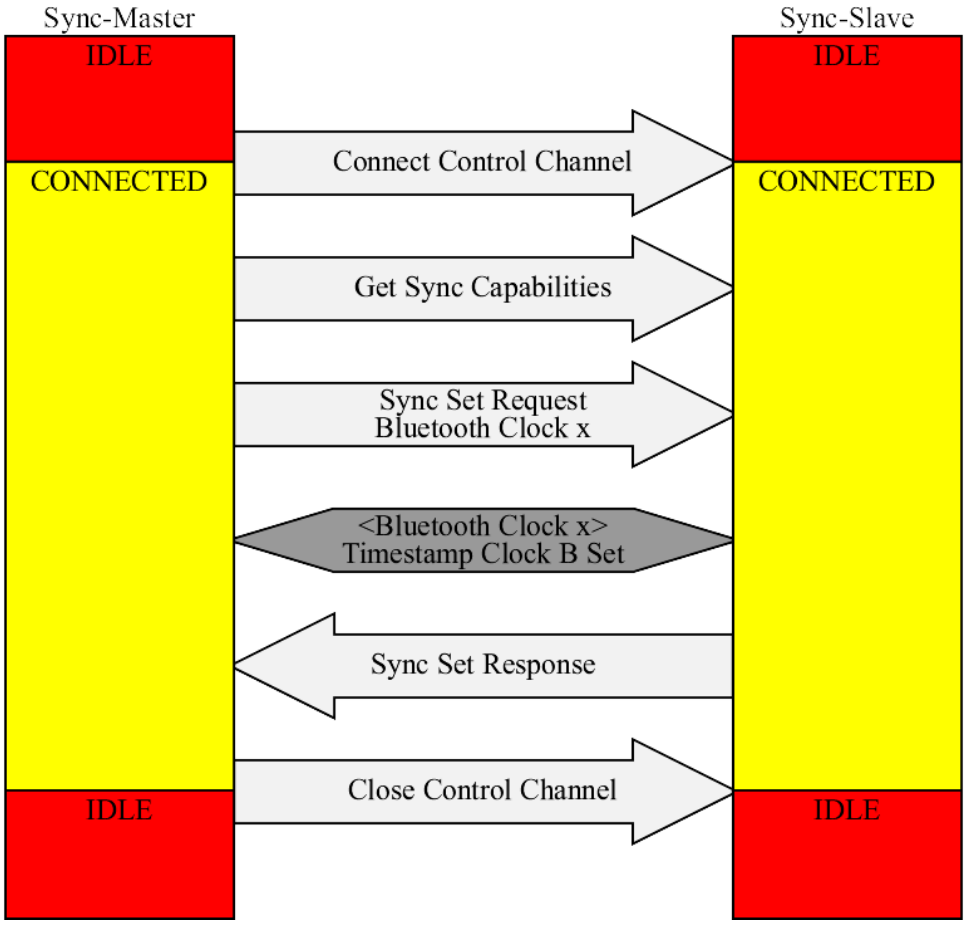

Figure 6.9: Protocol Transaction Example - Clock Synchronization - Simple Case

6.2.2 Clock Synchronization - Standard Case

This diagram shows a standard clock synchronization interaction, in which the Sync-Master (*Source*) initiates the connection, confirms the synchronization capabilities of the Sync-Slave (*Sink*), then requests a clock set event for a time in the near future (when the shared Bluetooth master clock reaches value x). The connection then proceeds as usual. When the Bluetooth master clock reaches time x, B sets its Time-Stamp Clock to the value requested in the "Set" request. This example shows that CSP commands can operate independently of MCAP standard commands.

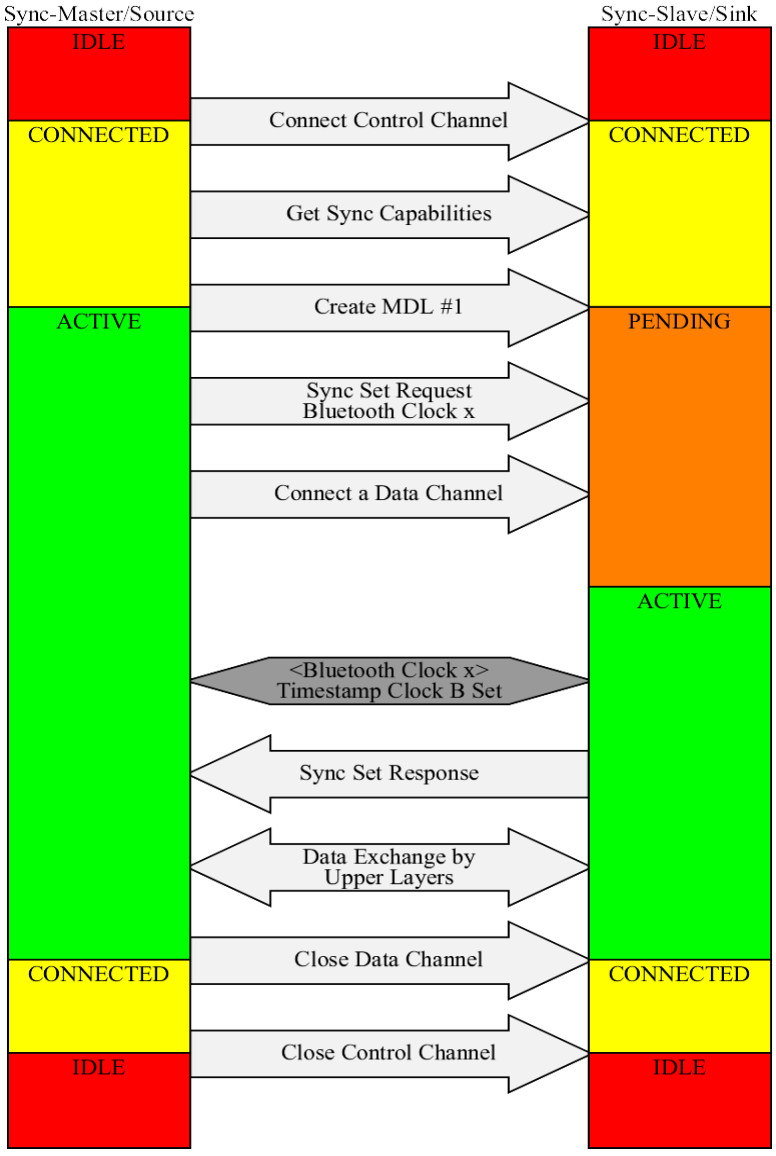

Figure 6.10: Protocol Transaction Example - Clock Synchronization - Standard Case

6.2.3 Clock Synchronization - Update Request Case

This diagram shows an example of a request for updates. Notice that clock synchronization requests are not affected by whether or not a Data Channel is open. In this case, the Data Channel was closed before the Control Channel, so the updates continued until the Control Channel was closed.

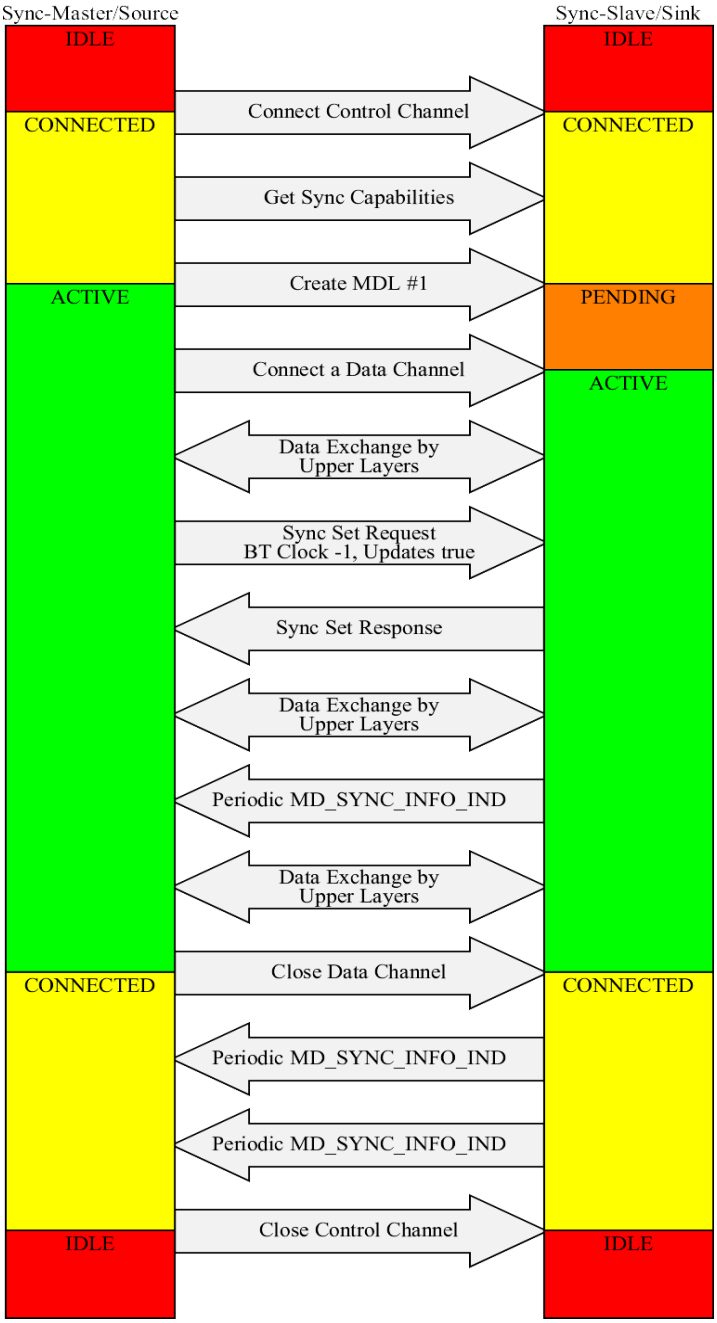

Figure 6.11: Protocol Transaction Example - Clock Synchronization - Update Request Case

7 References

References below are normative unless otherwise specified.

[1] Bluetooth Core Specification version 2.0 + EDR or later versions of the Bluetooth Core Specification

[2] Bluetooth SIG member web site, Bluetooth Assigned Numbers, http://www.bluetooth.org

[3] IEEE Standards Style Manual, http://standards.ieee.org/guides/style/2000Style.pdf

[4] 1003.1-2001/COR 1-2002 IEEE Standard for Information Technology - Portable Operating System Interface (POSIX) (Informative)

[5] Bluetooth Core Specification 2.0 + EDR or 2.1 + EDR with Volume 3, Part A of Core Specification Addendum 1 or later versions of the Bluetooth Core Specification

8 List of Figures

9 List of Tables

10 Appendix A: Acronyms and Abbreviations

Abbreviation or Acronym	Meaning
CE	Computation Engine
CSP	Clock Synchronization Protocol
L2CAP	Logical Link Control and Adaptation Protocol
MCAP	Multi-Channel Adaptation Protocol
MCL	MCAP Communications Link
MDEP	MCAP Device End Point
MDL	MCAP Device Link
MTU	Maximum Transmission Unit
PDU	Protocol Data Unit
PSM	Protocol/Service Multiplexer
QoS	Quality of Service
SAR	Segmentation and Reassembly
SDP	Service Discovery Protocol