

## Document Information Page

DO NOT DELETE

(This page contains hidden fields. To display the hidden text, choose Tools/Options/View and check Hidden Text or All (Outils/Options/Affichage, then check Texte masqué or Toutes).

/// Do not delete the following lines

**DOCUMENT INFORMATION FIELDS** (used to generate the title page, Document History, and page headers and footers):

To enter your document information:

1. First update the Document History page by inserting a new row in the table just above the last row, and retyping the information from the previous row.
2. Press ALT+F9 to display field codes (or choose Tools/Options/View/Field Codes -- français : Outils/Options/Affichage/Codes de champs).
3. For each field, enter your information after the word SET nnn (français: DEFINIR nnn) between quotes " ".
4. Press ALT+F9 again to hide field codes (or choose Tools/Options/View and remove the check from the Field Codes option -- français : Outils/Options/Affichage/Codes).
5. If certain fields are not updated after having pressed F9 (such as the page headers and footers), then choose File/Print Preview. You may need to repeat steps 1 and 3 again before the changes take effect.

In the table below, the hidden fields are in the right-hand column. If you cannot see them, press ALT+F9 or choose Tools/Options/View/Field Codes -- français : Outils/Options/Affichage/Codes):

Document Information Fields	Information for This Document	Press ALT+F9 and Enter Document Information Here
Document Authors:	AtosEuronextDocs, WConrad	
Document Issuer:	AtosEuronext	
Project Name 1 (only if 2 lines needed on title page):		
Project Name 2:		
Group or Client Associated with the Project:		
Document Title (55 characters maximum):	MMTP Technical Specifications	
Short Document Title (42 characters maximum):	MMTP Technical Specifications	
Document Subtitle (if any):		

Software Version:	2.14	
Document Version:	2.14_C	
Document Date:	April 2003	
Document Status:	FINAL	
Reviewed by:		
Date:		
Approved by:		
Date:		

**// Do not delete the preceding lines.**

# MMTP TECHNICAL SPECIFICATIONS

**Software Version 2.14**

**Document Version 2.14\_C (English)**

**April 2003**

---

---

## Document History

Document Version	Software Version/ Release	Document Author/Source	Description of Document Modifications
2.0D, 21 April 1999, Beta	2.10	EURONEXT <sup>SBF</sup> SA	Add refusal reason codes to CONX-NACK and DCNX-REQ. Change field types for Admin Data in DATA-MSG. M0 field added for shared access. Modify SYNC-ACK . Compatibility: HUB: Version R1-00-C and higher CAP: Version R1-00-C and higher SDK API MMTP: Version 2.10 and higher Note: Version 2.09 of the protocol and of the SDK API are still accepted.
2.0D, 25 May 25 1999, Final	2.10	EURONEXT <sup>SBF</sup> SA	Add START-NACK event to "MMTP out path" automaton
2.0E, 7 June 1999, Beta	2.11	EURONEXT <sup>SBF</sup> SA	Enhance the description of Route Data field (header type: E0). Add a new recovery mode in START-REQ Add a new primitive PRSC-MSG
2.0E, 28 June 1999, Final	2.12	EURONEXT <sup>SBF</sup> SA	Enhance the description of ERR-IND <b>Compatibility:</b> HUB: Version R4-01 and higher CAP: Version R4-01 and higher SDK API MMTP: Versions 2.11 and higher <b>Note:</b> It is recommended that you upgrade applications that were developed using versions prior to 2.11.

2.14, January 2001 FINAL (English)	2.14	AtosEuronext, Wconrad	Added: - Refusal Reason 07 to Section 5.4 - Disconnection Reason 09 to Section 5.5 - Error Code 9 in table, plus a Note, to section 5.13  Modified in table in section A.1: - <i>ATT-CNX</i> changed to <i>IDLE</i> in ERR-IND row + <i>ATT-CNX</i> column - Added a new ERR-IND row specific for Error Code 9 case
2.14A, February 2001 FINAL (English)	2.14	AtosEuronextDocs, Wconrad	Updated, corrected and reinforced info on admin data types in Section 5.7 and Appendix B. Removed admin data type CF, added admin data type G0.
2.14B, January 2002 FINAL (English)	2.14	Philippe Planchon	Corrected START-ACK: next sequence must be > 0 Added new field in special G0 admin data
2.14_C, April 2003 FINAL (English)	2.14	Philippe Planchon	Made minor corrections regarding ERR-IND.
Document Filename: PRD_MMTP2.14_TechSpec_20030428(v2.14_C).doc			
Save Date: 2003-04-28		Print Date: 2003-04-28	
Delivery Date: 2003-04-28			

**Note on the document version number:** For our product documents, usually the document has the same version number as the software that it is describing. For example, *ABC User's Guide*, Version 2.0 describes ABC software Version 2.0.

If later versions of a document are created for the *same* software version, the document version number has *\_A*, then *\_B* added to the end. For example, if later versions of *ABC User's Guide* are created for ABC software Version 2.0, the document version numbers are Version 2.0\_A, Version 2.0\_B, and so on.

---

## ***Send Us Your Comments***

AtosEuronext SA welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the revision process.

If you find any errors or have any other suggestions to improve the document quality and clarity, please indicate the chapter and page number (if available).

Please send comments to:

AtosEuronext SA  
Exchanges Products Division  
Quality Department  
6-8 boulevard Haussmann  
75009 Paris  
FRANCE

---

# Table of Contents

---

<b>1. Introduction .....</b>	<b>1</b>
1.1 Purpose .....	1
1.2 Document Scope .....	2
1.3 Related Documentation .....	3
<b>2. Main Characteristics .....</b>	<b>4</b>
2.1 HUB Architecture .....	4
2.2 TCP/IP Transport Protocol .....	6
2.3 Access Points .....	6
2.4 MMTP Sessions .....	7
2.5 Sequence Control and Acknowledgements .....	7
2.5.1 Sequence Checking .....	7
2.5.2 Acknowledgements .....	7
2.6 Restart Management .....	8
<b>3. Protocol Implementation .....</b>	<b>9</b>
3.1 Base-level Protocol .....	9
3.2 Optional Functions .....	9
3.3 List of Primitives .....	10
<b>4. Examples of MMTP Sessions .....</b>	<b>11</b>
4.1 Typical Exchange Session Using the Base-level Protocol .....	11
4.2 Standard Exchange Session with Acknowledgement .....	12
4.3 Typical Exchange Session with Retransmission .....	13
<b>5. Description of Message Primitives .....</b>	<b>14</b>
5.1 Standard Message Header .....	14
5.2 CONX-REQ: Client Connection Request .....	14
5.3 CONX-ACK: Connection Acceptance by the HUB .....	15
5.4 CONX-NACK: Connection Refusal by the HUB .....	16
5.5 DCNX-REQ: Disconnection Request .....	16
5.6 DCNX-ACK: Disconnection Request Acknowledgement .....	17
5.7 DATA-MSG Data Transmission .....	18
5.7.1 Message Primitive Format .....	18
5.7.2 Administrative Data Main Principles .....	19
5.7.3 Standard Administrative Data Fields .....	20
5.7.4 E0 Admin Data Type .....	21
5.7.5 E1 Admin Data Type .....	22

5.7.6	M0 Admin Data Type .....	23
5.7.7	MMTP OUT Path - Implementation Recommendation.....	24
5.7.8	MMTP IN Path - Implementation Recommendation.....	24
5.8	START-REQ: Transmission/Retransmission Request.....	24
5.9	START-ACK: Transmission/Retransmission Request Acknowledgement .....	25
5.10	START-NACK: Transmission/Retransmission Request Refusal .....	25
5.11	SYNC-REQ: Acknowledgement Request by Data Source.....	26
5.12	SYNC-ACK: Acknowledgement by Data Receiver .....	26
5.13	ERR-IND: Error Indication by Data Receiver.....	27
5.14	SRVC-MSG: Service Message.....	28
5.15	PRSC-MSG: Heartbeat Message.....	28
5.16	Direction of Protocol Message Primitive Transmission.....	29
<b>A.</b>	<b>Client Automata - MMTP IN Path and MMTP OUT Path .....</b>	<b>30</b>
A.1	Client MMTP IN Path .....	31
A.2	Client MMTP OUT Path .....	33
<b>B.</b>	<b>Special Administrative Data Formats.....</b>	<b>34</b>
B.1	A1 Admin Data Type.....	34
B.2	G0 Admin Data Type.....	36

# 1. Introduction

---

## 1.1 Purpose

AtosEuronext has created a standard protocol to simplify and improve access to trading and clearing services. The protocol, called MMTP (Market Message Transfer Protocol), allows client order/instruction entry applications to communicate with central exchange applications (e.g. trading applications) via the HUB Routing Engine.

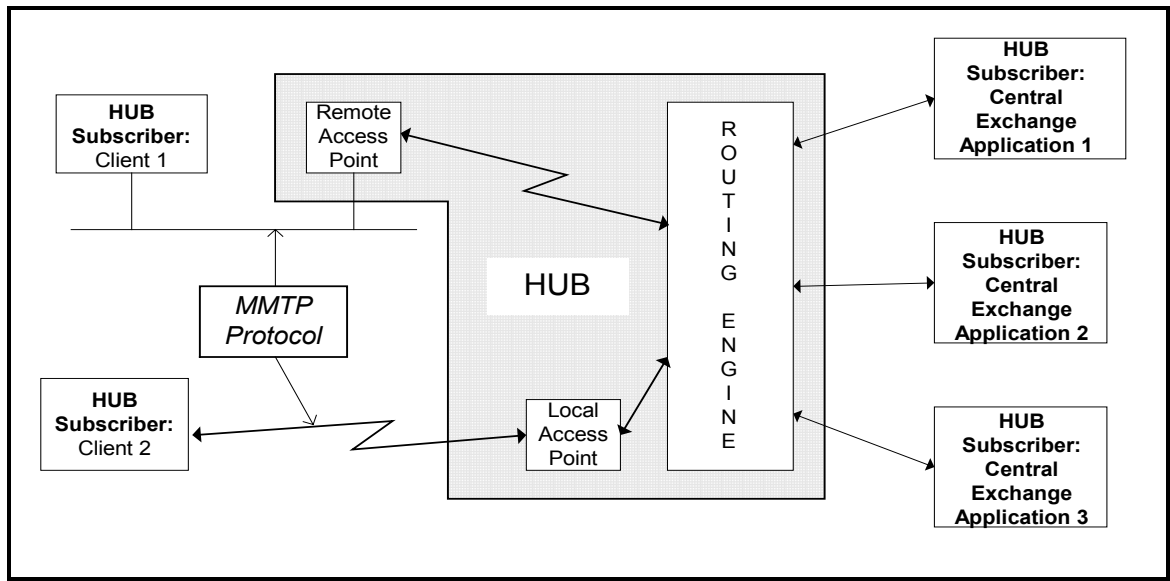
MMTP is adapted to all kinds of message mode dialogs, between:

- Client order entry applications and the HUB
- Client market data applications and the HUB
- Central exchange applications and the HUB

The MMTP protocol offers several advantages. It:

- Uses TCP/IP transport layers
- Guarantees transmission security
- Facilitates reconnection following disconnection
- Provides access to many applications without requiring addressing
- Transmits all market messages including those using formats such as SWIFT and TREX

Figure 1: MMTP Protocol



## 1.2 Document Scope

This document is designed to provide a detailed description of the MMTP protocol. Topics covered include:

- Chapter 2: The main characteristics of the MMTP protocol
- Chapter 3: The implementation of the protocol – levels, optional functions, revisions
- Chapter 4: Sample MMTP sessions
- Chapter 5: Detailed description of the different primitives used
- Appendix A: MMTP client automata for the MMTP IN and MMTP OUT paths

## 1.3 Related Documentation

This document intended to provide developers detailed information regarding the MMTP protocol. For general information regarding the HUB Open Interface solution, refer to:

- HUB Open Interface Solution: General Information Guide:

Overview of the HUB and its components: MMTP, CAP, HUB Server, HUB interface to the Trading Engine, Market Data Dissemination architecture, Infrastructure specifications. Includes a functional overview of trading principles and operations.

- HUB Technical Glossary: Hardware and Software Components:

Glossary containing definitions of the major terms and concepts used in the HUB Open Interface solution.

For more detailed documentation on the MMTP API, refer to:

- MMTP API Developer's Guide:

C function specifications, programming examples, session management, message logging

## **2. *Main Characteristics***

---

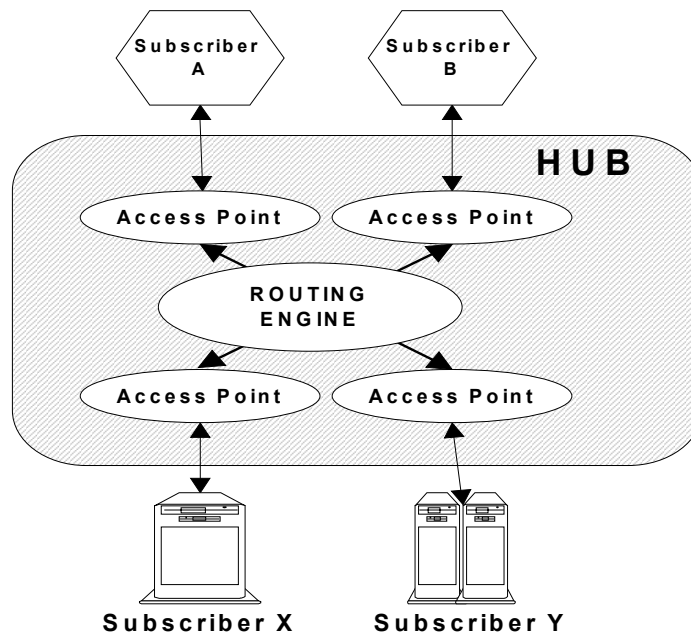
This chapter describes the main features of the MMTP protocol:

- HUB architecture
- TCP/IP transport protocol
- Access points
- MMTP sessions
- Sequence control and acknowledgements
- Restart management

### **2.1 HUB Architecture**

The following diagram illustrates the HUB architecture:

Figure 2: HUB Architecture



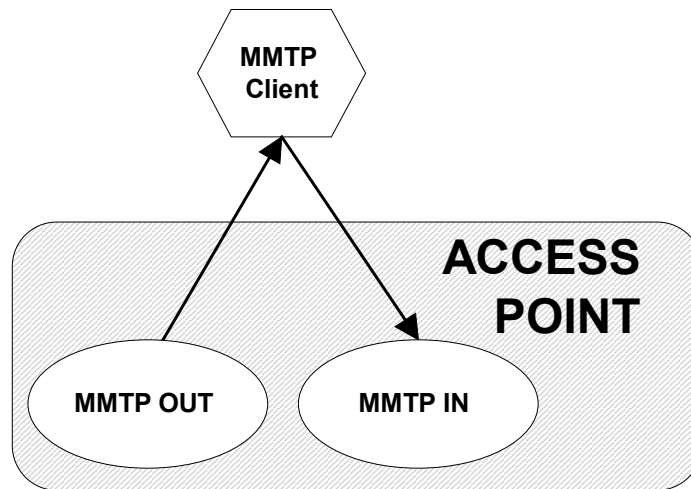
HUB transmits messages between HUB subscribers. A *HUB subscriber* is a client application running a member firm application or a central exchange application that is connected to the HUB.

- Subscribers connect to the HUB via an *access point*, a software module that provides the interface between a Certified Access Point (CAP) and the HUB, or between the HUB and the central exchange engines
- The MMTP protocol is used for dialog between the access point and the subscriber.
- HUB subscribers are managed by HUB members. A *HUB member* is a purely logical concept that groups several closely related physical entities. Members can be client application entities, central entities, or collect networks. A *collect network* is a service provider that manages the multiplexing of market message flows to supply an access solution to several members.
- The *Routing Engine* is a HUB Server component providing message handling and routing functions.

The HUB subscriber always initiates the connection. In MMTP protocol terms, the subscriber is the *MMTP client*. The following terms will be used in this document:

- *MMTP client* for the HUB subscriber
- *MMTP server* for the access point

Figure 3: MMTP Client



To optimize performance, two separate paths are used for data exchange:

- MMTP client to HUB: **MMTP IN** → application message feed transmitted by the MMTP client to the MMTP server.
- HUB to MMTP client: **MMTP OUT** → application message feed from the MMTP server to the MMTP client.

Furthermore, both the MMTP IN and MMTP OUT paths are bi-directional. For example, the MMTP server can reply to a message from the MMTP client on the MMTP IN path.

## 2.2 TCP/IP Transport Protocol

The TCP/IP protocol was chosen for its flexibility:

- It supports different types of network equipment such as point-to-point lines and Ethernet, and,
- It is available for all types of platforms

## 2.3 Access Points

The HUB subscriber connects to the system via an access point which uses the MMTP protocol. There are two types of access points:

- The *CAP* or Certified Access Point resides at the member site and provides secure communications to the HUB via a wide area network. The CAP also performs compression, encryption, and session management functions for transmitting messages between the order message/market data applications and the HUB. The CAP is specially adapted for the connection of client order entry and market data applications to the HUB.
- The local access point allows server applications to connect to the HUB via a local area network.

## 2.4 MMTP Sessions

The MMTP protocol implements dialogs in connected mode. A client application must establish an MMTP session before being able to communicate via MMTP. An MMTP session involves three phases:

- Connection/authentication
- Transmission and retransmission
- Disconnection

## 2.5 Sequence Control and Acknowledgements

Each DATA type MMTP message must be assigned a sequence number that increases logically. The sequence number allows:

- The receiver to check that it has received all messages in the correct order
- The sender to request the acknowledgement of a specific message

At the start of each MMTP session, the sender selects the sequence number to be assigned to the first message of the session to be transmitted, and then communicates it to the receiver. A sequence number can be used several times in consecutive sessions. It is not specific to a single message, but rather to a single session.

### 2.5.1 Sequence Checking

When a receiver detects an error in the sequence, it alerts the sender to the problem, indicating the sequence number received and the number expected.

- If the number received corresponds to a message already received, the new message is ignored.
- If the sequence number received is higher than the number expected, the receiver alerts the sender and then initiates a new connection.

### 2.5.2 Acknowledgements

With MMTP, the systematic acknowledgement of messages is not mandatory. However, if one party in the dialog requests it, specific acknowledgement can be sent:

- The sender can request the last sequence number received by the receiver, if it so wishes
- The receiver must reply, indicating the last sequence number it processed correctly

This message acknowledgement mechanism can be used to manage application windowing.

## 2.6 Restart Management

The TCP transport protocol guarantees the integrity of the data exchanged by the MMTP client and the HUB. However, in the event of an accidental disconnection, the information in the transmission or reception buffers may be lost causing messages to be lost.

Message delivery is guaranteed by using the following principles:

- Each message has a unique identifier, the **message ID**, assigned by the sender (see 5.7).
- At the beginning of each session, the receiver communicates the last **message ID** received to the sender.

---

## 3. Protocol Implementation

---

This chapter briefly covers how the MMTP protocol is implemented, describing the base-level protocol and optional functions. It also includes a list of message primitives.

### 3.1 Base-level Protocol

The base-level protocol is used for data exchange between the MMTP client and the HUB. It provides the following elementary functions:

- Client connection request acknowledgement or refusal by the HUB
- Data transmission
- Data retransmission
- Disconnection request and disconnection request acknowledgement

The MMTP client identifies itself with the connection request. It contains the client's HUB subscriber ID and authentication data. The MMTP server accepts or refuses the connection based on the administrative data stored in the HUB.

Sequence number errors are indicated in the data transmission. Data retransmission allows the receiver of a message feed to request the retransmission of all or some of the messages received (see Section 2.6, Restart Management).

Disconnection requests and disconnection request acknowledgements allow the receiver to indicate the last sequence number received and accepted.

### 3.2 Optional Functions

When the MMTP client initiates the connection, the MMTP server can specify additional functions that modify the protocol's behavior. The MMTP server then imposes those functions on the client when the connection is acknowledged.

At present, the use of optional functions is reserved for server applications.

Certified Access Points systematically use Session Configuration option 2. Refer to Section 5.2, CONX-REQ: Client Connection Request for details.

### 3.3 List of Primitives

The following table lists the message primitives used by the protocol:

Primitive	N°	Description
CONX-REQ	10	Client connection request
CONX-ACK	11	Connection acceptance by the HUB
CONX-NACK	12	Connection refusal by the HUB
DCNX-REQ	13	Disconnection request
DCNX-ACK	14	Disconnection request acknowledgement
START-REQ	20	Transmission/retransmission request
START-ACK	21	Transmission/retransmission request acknowledgement
START-NACK	22	Transmission/retransmission request refusal
DATA-MSG	23	Data transmission
SYNC-REQ	24	Acknowledgement request by data source
SYNC-ACK	25	Acknowledgement by data receiver
ERR-IND	90	Error indication by data receiver
SRVC-MSG	93	Service message
PRSC-MSG	99	Heartbeat message

For a detailed description of these primitives, see Chapter 5, Description of Message Primitives.

## 4. Examples of MMTP Sessions

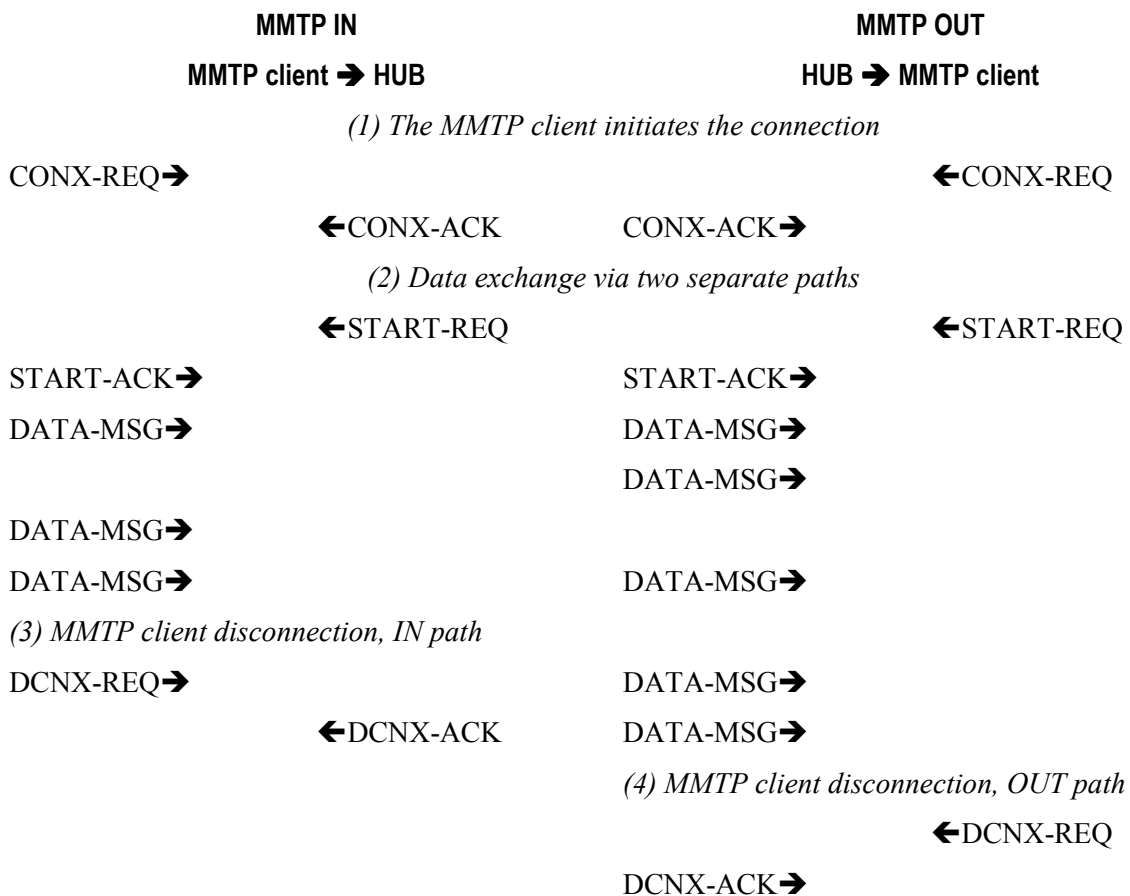
This chapter illustrates three typical MMTP sessions:

- The first example uses the base-level protocol only
- The second example uses the base-level protocol plus session acknowledgement
- The third example includes retransmission

For each session, the dialog is illustrated by a table with two columns:

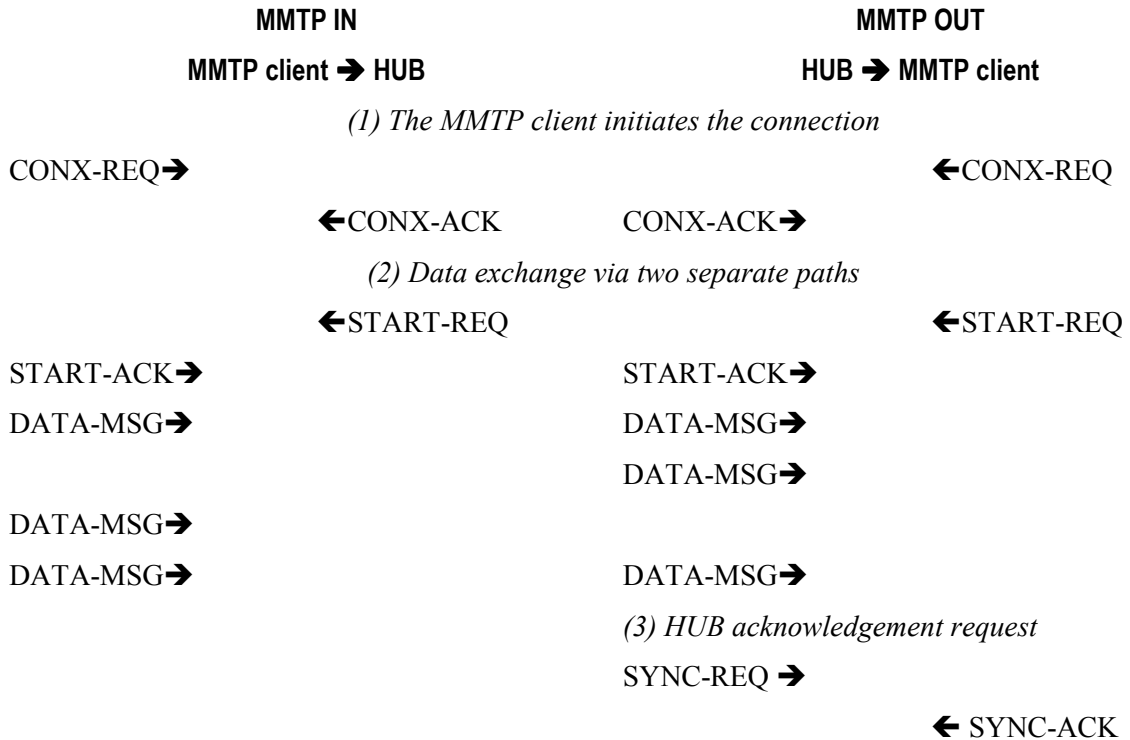
- The first column represents the MMTP IN path, for messages sent from the MMTP client to the HUB.
- The second column represents the MMTP OUT path, for messages sent from the HUB to the MMTP client.

### 4.1 Typical Exchange Session Using the Base-level Protocol

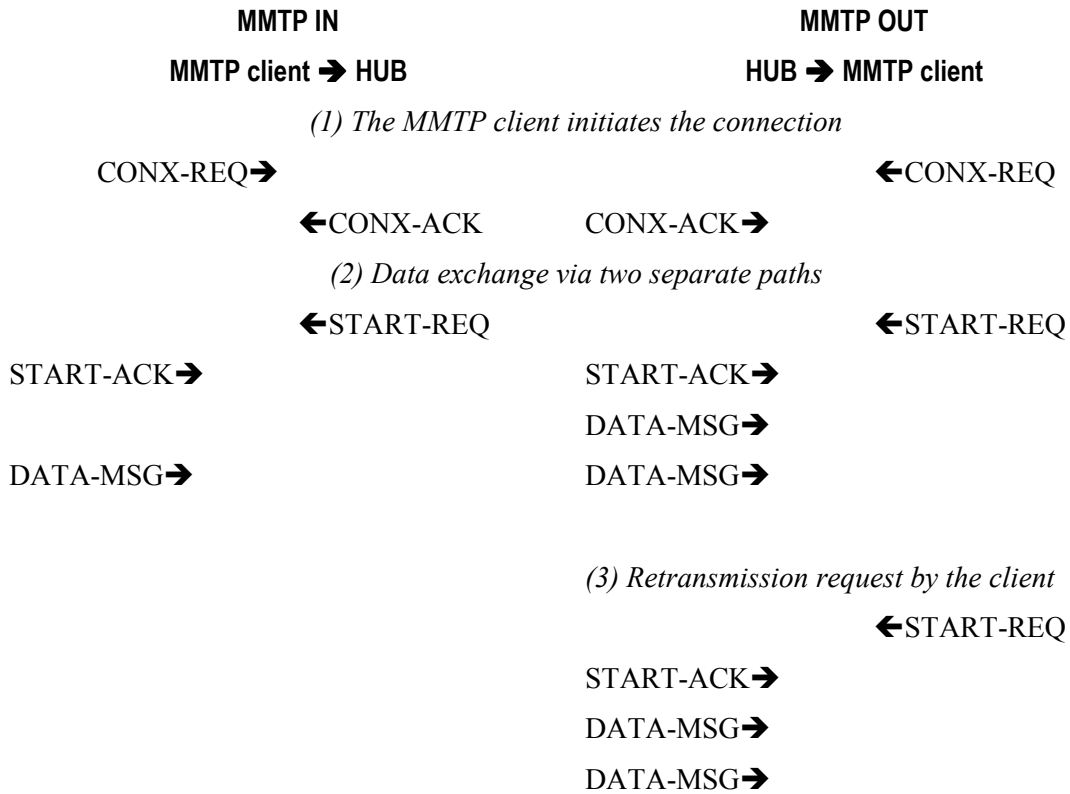


Note that the two columns in this table are independent, because the MMTP IN and MMTP OUT paths operate independently of one another. For example, part 3 of the table contains a DATA-MSG on the MMTP OUT side while a DCNX-REQ has been sent on the MMTP IN side. This is because the MMTP OUT side (e.g. HUB to CAP) could still be sending messages to a CAP process, while on the MMTP IN side (e.g. CAP to HUB), another CAP process requests and receives a disconnection.

## 4.2 Standard Exchange Session with Acknowledgement



### 4.3 Typical Exchange Session with Retransmission



## 5. Description of Message Primitives

This chapter provides a detailed description of each MMTP message primitive.

### 5.1 Standard Message Header

All the protocol messages have the same header. It allows easy identification of messages in the character feed transmitted by TCP/IP and enables the rapid rectification of transmission errors.

Field name	Offset	Length	Type	Value	Comment
STX	0	1		02	Value in hexadecimal
Primitive Length	1	4	num.	=N+6	Total length of the message including STX and ETX
DATA	5	N			Body of the message
ETX	N+5	1		03	Value in hexadecimal

Excluding the STX and ETX delimiters, all information in the messages is in printable ASCII code: letters, numbers, punctuation marks and mathematical symbols.

Numerical data are inserted on the right and padded with zeros on the left.

Other data are inserted on the left and padded with spaces on the right.

### 5.2 CONX-REQ: Client Connection Request

MMTP clients use this primitive to identify themselves when they connect to the HUB and the HUB uses it to check the whether the client is authorized to connect.

Field name	Offset	Length	Type	Value	Comment
Primitive Identifier	5	2	num.	10	
HUB Subscriber ID	7	11			
Protocol Version	18	4	num.		
Session Configuration	22	16	num.		
Authentication Data	38	8			

Message size: 47

**Primitive Identifier:** Identifies the primitive

**Hub Subscriber ID:** Assigned to the client when the connection is established. It is used to check the connection's local and remote addresses. If the MMTP client has an order entry application, its order entry application number is the same as its HUB Subscriber ID.

**Protocol Version:** Checks that the client version of the protocol is compatible with the versions accepted by the MMTP server. The MMTP client must not use a protocol version later than the latest version accepted by the server.

The version of the protocol described in this document is 2.14.

**Session Configuration:** This field consists of 16 characters indicating which dialog options are used. Each character corresponds to an option and can have one of two values:

- 0 if the option is not used
- 1 if the option is used

The following options are available:

Option	Description
1	The HUB server is sending encrypted data
2	Disconnection following a sequence error
3 to 16	For future use

**Warning:** When a Certified Access Point sends a CONX-REQUEST to the HUB Server, it sets option 2 to ON (a value of 0100000000000000).

**Authentication Data:** This field contains a password associated with the Hub Subscriber ID field and is managed by the HUB.

**Note:** Two successive attempts at connection must be separated by a ten-second delay, otherwise the request will be refused (see Section 5.4, CONX-NACK: Connection).

### 5.3 CONX-ACK: Connection Acceptance by the HUB

The HUB uses this primitive to validate the connection.

Field name	Offset	Length	Type	Value	Comment
Primitive Identifier	5	2	num.	11	
Session Configuration	7	16	num.		

Message size: 24

**Primitive Identifier:** Identifies the primitive

**Session Configuration:** Contains the values for the message exchange session, at the time of connection. The values in this field are determined by the client's request and constraints of the HUB access point. The access point indicates to the client all the parameters to be taken into account during the data exchange. For details on the Session Configuration field, see Section 5.2, CONX-REQ: Client Connection Request.

When the HUB Server replies to a CONX-REQ from a Certified Access Point, it uses the Session Configuration field to indicate its decision regarding the connection to the CAP. If the HUB decides to encrypt the data, it sets option 1 to ON (it returns a value of (a value of 1000000000000000)).

## 5.4 CONX-NACK: Connection Refusal by the HUB

The HUB uses this primitive to indicate connection refusal to MMTP clients.

Field name	Offset	Length	Type	Value	Comment
Primitive Identifier	5	2	num.	12	
Refusal Reason	7	2	num.		See values below

Message size: 10

**Primitive Identifier:** Identifies the primitive

**Refusal Reason:**

No.	Meaning
01	Client connection to the HUB is not authorized
02	HUB is not available for client connections
03	Incorrect identification (client identifier and/or password is incorrect)
04	Last request for connection is too recent (time delay not respected)
05	Incompatible version of the protocol
06	Invalid protocol options
07	Too many connections

## 5.5 DCNX-REQ: Disconnection Request

This primitive is used to end the connection.

Field name	Offset	Length	Type	Value	Comment
Primitive Identifier	5	2	num.	13	
Disconnection Reason	7	2	num.		See values below
Last Sequence	9	8	num.		Latest Sequence Number transmitted/accepted

Message size: 18

**Primitive Identifier:** Identifies the primitive

**Disconnection Reason:**

No.	Meaning
01	Normal disconnection
02	HUB administration request for disconnection
03	Abnormal disconnection
04	Interruption of the session by the HUB
05	START-ACK not received or the START-ACK message ID does not correspond to the data requested in the START-REQ
06	Abnormal disconnection due to emission of the last message in the day
07	Incorrect encryption key used
08	Admin Data type not authorized
09	Access control list violation
10	Encryption unavailable
99	Last message sent

Last Sequence:

- If the request is made by the data source, the Last Sequence is the sequence number for the last DATA-MSG transmitted.
- If the request is made by the data receiver, the Last Sequence is the sequence number for the last DATA-MSG received.

## 5.6 DCNX-ACK: Disconnection Request Acknowledgement

Field name	Offset	Length	Type	Value	Comment
Primitive Identifier	5	2	num.	14	
Last Sequence	7	8	um		Last sequence number transmitted/received

Message size: 16

**Primitive Identifier:** Identifies the primitive

**Last Sequence:**

- If the disconnection is accepted by the data source, the Last Sequence is the last sequence number transmitted.
- If the disconnection is accepted by the data receiver, the Last Sequence is the last sequence number received and accepted.

## 5.7 DATA-MSG Data Transmission

### 5.7.1 Message Primitive Format

MMTP clients and the HUB use this primitive to exchange data.

Field name	Offset	Length	Type	Value	Comment
Primitive Identifier	5	2	digits	23	Identifies the primitive
Sequence Number	7	8	digits		Assigned by the sender as follows: For the first DATA-MSG received following the START-ACK, the Sequence Number is indicated by the Next Sequence field of the START-ACK Thereafter the Sequence Number increases by 1 for each new DATA-MSG
Admin Data Length	15	4	digits	N	Indicates the length of the Admin Data field
Business Data Length	19	4	digits	M	Indicates the length of the Business Data field
Admin Data	23	N			N < 256 Contains administrative data concerning the Business Data field (see details below)
Business Data	23+N	M			M < 9500 For a description of this data, refer to the documents accompanying the application servers (such as the Trading Engine or the Clearing Engine) accessible via the HUB.

The size of this kind of message is  $24 + N + M$ .

Note: All messages received or issued by the HUB using the MMTP protocol are sent using the admin data type.

### 5.7.2 Administrative Data Main Principles

The administrative data in a DATA-MSG data transmission primitive contains information on message contents or message destination. Administrative data is structured using one of the following pre-defined types:

- **E0**: External type 0
- **E1**: External type 1
- **M0**: Shared type 0
- **A1**: Application type 1
- **G0**: Gateway type 0

**Note:** Type **A1** is used only by applications that access the HUB via Application Access Points, and type **G0** for dialog between two HUBs. For more details on the A1 and G0 types, refer to Appendix B.

For the remainder of this document, the term *admin data type* is used to describe the special format for administrative data.

The **Admin Data** field contains the following:

- A message identifier
- Additional technical and functional data.

The field is structured, of variable length and format. It is made up of three mandatory fields plus additional optional fields.

The HUB applies the following rules when handling administrative data:

- Messages entering the HUB:

A message is received along with its envelope. Certain fields in the envelope may be used for routing purposes: they are defined in the HUB as fields of type Technical. The envelope is stored in the HUB in an internal format.

- Messages leaving the HUB:

The envelope the HUB uses to send a message may be different from the envelope in which it was received. The envelope used for transmission is defined in the HUB administration database.

### 5.7.3 Standard Administrative Data Fields

Admin Data is made up of four mandatory fields plus additional optional fields.

Field name	Offset	Length	Type	Value	Comment
Type	23	2	char.		<p>Indicates the admin data type.</p> <p>This field may have one of the following values:</p> <ul style="list-style-type: none"> <li>➤ E0 and E1: the only data types accepted by the CAP. The CAP accepts data types E0 and E1 on the MMTP IN path, and E1 exclusively on the MMTP OUT path. Refer to sections 5.7.4 E0 Admin Data Type and 5.7.5 E1 Admin Data Type.</li> <li>➤ M0: the only data type accepted by MAPs. Refer to section 5.7.6 M0 Admin Data Type.</li> <li>➤ A1: data type used only by applications that access the HUB via Application Access Points. Refer to Appendix B.1 A1 Admin Data Type</li> <li>➤ G0: data type used only for communication between two HUBs. Refer to Appendix B.2 G0 Admin Data Type</li> </ul>
MsgId	25	24	char.		<ul style="list-style-type: none"> <li>• Message identifier assigned by the sender.</li> <li>• <b>MsgId is UNIQUE for each HUB subscriber and for each message sent</b></li> </ul>
SendTime	49	12	digits		<ul style="list-style-type: none"> <li>• Send timestamp</li> <li>• Format: MMDDHHmmscc</li> <li>• Date and time of message transmission.</li> </ul>
ReceiptTime	61	12	digits		<ul style="list-style-type: none"> <li>• Receive timestamp</li> <li>• Format: MMDDHHmmscc</li> <li>• Date and time of message reception.</li> </ul>
Optional fields	73	N			<ul style="list-style-type: none"> <li>• Vary according to the type of field determined by Admin Data Type</li> </ul>

**Note:** Shaded rows indicate generic information present in all the different Admin Data Types

### 5.7.4 E0 Admin Data Type

The E0 Admin Data Type is used exclusively for data messages transmitted via the CAP.

Field name	Offset	Length	Type	Value	Comment
Type	23	2	char.	E0	Standard field - Refer to Section 5.7.3
MsgId	25	24	char.		Standard field - Refer to Section 5.7.3
SendTime	49	12	digits		Standard field - Refer to Section 5.7.3
ReceiptTime	61	12	digits		Standard field - Refer to Section 5.7.3
DeliveryTimeout	73	6	digits		<ul style="list-style-type: none"> <li>• Format: HHmmss</li> <li>• This field is a timestamp, NOT a delay</li> <li>• <b>For future use</b></li> </ul>
RouteData	79	11	char.		<ul style="list-style-type: none"> <li>• Routable field</li> <li>• May be used by the HUB for routing data: ROUTE_DATA.</li> <li>• Additional data for routing messages:               <ul style="list-style-type: none"> <li>➤ for the MMTP IN path: Not meaningful unless specified</li> <li>➤ for the MMTP OUT path: Filled with the mnemonic identifier of the HUB member which originally sent the message (8 left-justified characters)</li> </ul> </li> </ul>
Filler	90	5	char.		Characters to fill E0 administrative data to its defined size (see Note below).

**Note:** Size of E0 administrative data: 72

### 5.7.5 E1 Admin Data Type

The E1 Admin Data Type is used exclusively for data messages transmitted via the CAP.

Field name	Offset	Length	Type	Value	Comment
Type	23	2	char.	E1	Standard field - Refer to Section 5.7.3
MsgId	25	24	char.		Standard field - Refer to Section 5.7.3
SendTime	49	12	digits		Standard field - Refer to Section 5.7.3
ReceiptTime	61	12	digits		Standard field - Refer to Section 5.7.3
DeliveryTimeout	73	6	digits		<ul style="list-style-type: none"> <li>• Format: HHmmss</li> <li>• This field is a timestamp, NOT a delay</li> <li>• <b>For future use</b></li> </ul>
Filler	79	8	char.		Characters to fill E1 administrative data to its defined size (see Note below).

**Note:** Size of E1 administrative data: 64

### 5.7.6 M0 Admin Data Type

The M0 Admin Data Type is used exclusively for data messages transmitted via MAP access points.

Field name	Offset	Length	Type	Value	Comment
Type	23	2	char	M0	Standard field - Refer to Section 5.7.3
MsgId	25	24	char		Standard field - Refer to Section 5.7.3
SendTime	49	12	digits		Standard field - Refer to Section 5.7.3
ReceiptTime	61	12	digits		Standard field - Refer to Section 5.7.3
DeliveryTimeout	73	6	digits		<ul style="list-style-type: none"> <li>• Format: HHmmss</li> <li>• This field is a timestamp, NOT a delay</li> <li>• <b>For future use</b></li> </ul>
RouteData	79	11	char		<ul style="list-style-type: none"> <li>• Additional routing data: <ul style="list-style-type: none"> <li>➢ for the <b>MMTP OUT</b> path: Filled with the mnemonic identifier of the HUB member which originally sent the message (8 left-justified characters).</li> <li>➢ for the <b>MMTP IN</b> path: Not meaningful unless specified</li> </ul> </li> </ul>
OnMember	90	8	char		<ul style="list-style-type: none"> <li>• Member ID: <ul style="list-style-type: none"> <li>➢ for the <b>MMTP OUT</b> path: Identifies the member on whose behalf the message is being sent</li> <li>➢ for the <b>MMTP IN</b> path: If filled, identifies the member to whom the message is being sent</li> </ul> </li> </ul>
Domain	98	1	char		<ul style="list-style-type: none"> <li>• Technical fields: <ul style="list-style-type: none"> <li>➢ for the <b>MMTP OUT</b> path: Not meaningful</li> <li>➢ for the <b>MMTP IN</b> path: Internal use only</li> </ul> </li> </ul>
Dest	99	11	char		<ul style="list-style-type: none"> <li>• Technical fields: <ul style="list-style-type: none"> <li>➢ for the <b>MMTP OUT</b> path: Not meaningful</li> <li>➢ for the <b>MMTP IN</b> path: Internal use only</li> </ul> </li> </ul>
Filler	110	41	char		Characters to fill M0 administrative data to its defined size (see Note below).

**Note:** size of M0 administrative data: 151

### 5.7.7 MMTP OUT Path - Implementation Recommendation

The MMTP client should save the MsgId field in the Admin Data field for each message received. At the beginning of each session, the client must be able to supply the MsgId value of the last message received.

### 5.7.8 MMTP IN Path - Implementation Recommendation

To simplify restart management, use the MsgId field in the Admin Data field to store the sequence of numbers for messages sent. When a START-REQ is received, the MsgId given in the START-REQ is compared with the MsgId fields in the messages sent to determine which messages to resend.

## 5.8 START-REQ: Transmission/Retransmission Request

The receiver uses this primitive to indicate to the sender that it must start the transmission of DATA-MSGs with the message indicated by the **Message ID**. This message must be sent before data can be received. The data source must reply to this message within a specified time period.

Field name	Offset	Length	Type	Value	Comment
Primitive Identifier	5	2	num.	20	
Message ID	7	24			

Message size: 32

**Primitive Identifier:** Identifies the primitive.

**Message ID:** The identifier of the last message received, 24 characters:

- If the **Message ID** field is blank, the sender must transmit the entire feed from the morning of the current day.
- If the **Message ID** is filled with START WITH NEXT MESSAGE! (24 characters: Four uppercase words separated by one space and ended by an exclamation mark), then the sender must resume the data transmission from the next message it receives. Thus all the awaiting messages are skipped and the real-time feed resumes with the next message.

**Note:** The START WITH NEXT MESSAGE! mode is only allowed through the MMTP OUT path of market data client applications.

## 5.9 START-ACK: Transmission/Retransmission Request Acknowledgement

This primitive is used to acknowledge a transmission request from the data source.

Field name	Offset	Length	Type	Value	Comment
Primitive Identifier	5	2	num.	21	
Next Sequence	7	8	num.		next sequence number transmitted, must be > 0
Message ID	15	24			

Message size: 40

**Primitive Identifier:** Identifies the primitive

**Next Sequence:** Indicates the first sequence number in the transmission feed to the receiver. This sequence number is chosen by the feed source at its convenience, but must not be zero.

**Message ID:** The same as the Message ID for the corresponding START-REQ.

## 5.10 START-NACK: Transmission/Retransmission Request Refusal

This primitive is used to refuse a transmission/retransmission request (START-REQ) from the sender.

Field name	Offset	Length	Type	Value	Comment
Primitive Identifier	5	2	num.	22	
Refusal Reason	7	2	num.		see below for details
Message ID	9	24			

Message size: 34

**Primitive Identifier:** Identifies the primitive

**Refusal Reason:**

No.	Meaning
01	START-REQ already requested but not acknowledged
02	no connection to the HUB exists
03	message specified by Admin Data not found

**Message ID:** The same as the corresponding START-REQ Message ID

## 5.11 SYNC-REQ: Acknowledgement Request by Data Source

This primitive is used to request the last correct sequence number received by the receiver in order to determine at which point to restart transmission.

Field name	Offset	Length	Type	Value	Comment
Primitive Identifier	5	2	num.	24	

Message size: 8

**Primitive Identifier:** Identifies the primitive

The sender determines the speed at which the primitive is transmitted.

From time to time the MMTP server sends a SYNC-REQ on the MMTP OUT path. On reception of a SYNC-REQ, the receiver must send a SYNC-ACK in reply.

## 5.12 SYNC-ACK: Acknowledgement by Data Receiver

This primitive is used to reply to an acknowledgement request (SYNC-REQ). The sequence number to be used is the last valid sequence number received. Unless the sequence error blocking option is activated, reception of the sequence numbers preceding the current number cannot be guaranteed.

Field name	Offset	Length	Type	Value	Comment
Primitive Identifier	5	2	num.	25	
Last Sequence	7	8	num.		Last sequence number accepted
Message ID	15	24	char		

Message size: 40

**Primitive Identifier:** Identifies the primitive

**Last Sequence:** The last DATA-MSG sequence number received by the receiver

**Message ID:** Copy of the Message ID field for the last DATA-MSG processed

### 5.13 ERR-IND: Error Indication by Data Receiver

This primitive is used to indicate abnormalities.

Field name	Offset	Length	Type	Value	Comment
Primitive Identifier	5	2	num.	90	
Error Code	7	2	num.		see below for details
Error Detail	9	2	num.		see below for details
Last Sequence	11	8	num.		
Refused Message Length	19	4	num.	N	does not include STX, ETX and primitive length
Refused Message	23	N			does not include STX, ETX and primitive length

Message size: 24+N

**Primitive Identifier:** Identifies the primitive

Error Code	Error Detail	Description
1	1	Message lost. Refusal of the message indicated
2	0	Message already received
3	N	Field number N is incorrect (N=0 for STX)
4	0	The message received is out of context
5	0	The message received is not valid
9	0	Acces control list violation

For error codes 2 to 5, the message and its sequence number are ignored.

For error code 9, the message is stored but will not be delivered to destination.

Error codes 3 and 5 can be used for other primitives besides DATA-MSG. They indicate a protocol error.

**Last Sequence:** The last DATA-MSG sequence number accepted by the receiver. This number can be used to define a point for restarting transmission.

**Refused Message Length:** The STX and ETX of the invalid message are not used in the error primitive to avoid confusion with the STX and ETX delimiting the ERR-IND primitive. If the invalid message cannot be returned, its length is 0.

## 5.14 SRVC-MSG: Service Message

This primitive is used to send service messages. The messages can be processed or ignored depending on the mandatory or optional nature of the protocol service.

Field name	Offset	Length	Type	Value	Comment
Primitive Identifier	5	2	num.	93	
Service Type	7	4	char.		see below for details
Service Data Length	11	4	num.	N	
Service Data	15	N	num.		see below for details

Message size: 16+N

**Primitive Identifier:** Identifies the primitive

**Service Type:** Identifies the service offered by the message.

**Service Data Length:** Indicates the data feed length. If no data follows the message, the length is "0000".

**Service Data:** Contains additional data. The content of the field depends on the type of service.

The following services are currently available:

- Service Type = PING: This optional service is used to verify that a partner is still present.  
Service Data: Contains the date and time of transmission in the format
- Service Type = PONG: This mandatory service is used to reply to a PING message.  
Service Data: The PING Service Data field is copied for the reply.

**Notes:**

- An MMTP session must never send a PONG Service Type except in reply to a PING Service Type.
- Each MMTP session MUST reply to a PING Service Type with a PONG Service Type.

## 5.15 PRSC-MSG: Heartbeat Message

This primitive is sent during periods of message inactivity, whatever the logical connection status (even before the logical connection is opened). It is sent every few seconds (for example every ten seconds). It is used to indicate that an MMTP session is still valid and that the connection has not been lost.

Field name	Offset	Length	Type	Value	Comment
Primitive Identifier	5	2	num.	99	

Message size: 8

**Primitive Identifier:** Identifies the primitive

## 5.16 Direction of Protocol Message Primitive Transmission

Primitive Direction	MMTP IN		MMTP OUT	
	Client → HUB	HUB → Client	Client → HUB	HUB → Client
CONX-REQ	✓		✓	
CONX-ACK		✓		✓
CONX-NAK		✓		✓
DCNX-REQ	✓	✓	✓	✓
DCNX-ACK	✓	✓	✓	✓
START-REQ		✓	✓	
START-ACK	✓			✓
START-NACK	✓			✓
DATA-MSG	✓			✓
SYNC-REQ	✓			✓
SYNC-ACK		✓	✓	
ERR-IND	✓	✓	✓	✓
SRVC-MSG	✓	✓	✓	✓
PRSC-MSG	✓	✓	✓	✓

Separate paths are used for all data exchange, including connection and disconnection.

## A. Client Automaton - MMTP IN Path and MMTP OUT Path

This Appendix provides tables listing the possible states of a connection between an MMTP client and an MMTP server, and showing how different events affect the state of the connection. The first table concerns the MMTP IN path and the second concerns the MMTP OUT path.

Table key:

- The first column lists the events
- The first row lists the current states
- The cells list the primitive sent (in capitals) and the resulting state (in italics).

<i>STATE</i>	<i>STATE 1</i>	<i>STATE 2</i>	<i>STATE 3</i>	<i>STATE 4</i>
Event				
Event 1		PRIMITIVE 1 <i>Resulting state</i>		
Event 2	PRIMITIVE 2 <i>Resulting state</i>		Action 1 <i>Resulting state</i>	PRIMITIVE 2 <i>Resulting state</i>

The events can be external to the client applications (coming from the HUB) or internal (data transmission request).

List of possible states:

- **IDLE**: Not connected
- **ATT-CNX**: Awaiting connection acceptance
- **ATT-START**: Awaiting transmission/retransmission request
- **CNX**: Connection established, transmission authorized
- **ATT-ACK (SYNC-ACK)**: Awaiting reply to an acknowledgement request (MMTP IN path only)
- **ATT-ACK (START-ACK)**: Awaiting reply to the transmission/retransmission request (MMTP OUT path only)
- **ATT-DCNX**: Awaiting disconnection acknowledgement

## A.1 Client MMTP IN Path

	<i>IDLE</i>	<i>ATT-CNX</i>	<i>ATT-START</i>	<i>CNX</i>	<i>ATT-ACK</i> ( <i>SYNC-ACK</i> )	<i>ATT-DCNX</i>
Connection request	<i>CONX-REQ</i> <i>ATT-CNX</i>					
<i>CONX-ACK</i>	<i>ERR-IND</i> <i>IDLE</i>	<i>ATT-START</i>	<i>ERR-IND</i> <i>ATT-START</i>	<i>ERR-IND</i> <i>CNX</i>	<i>ERR-IND</i> <i>ATT-ACK</i>	<i>ATT-DCNX</i>
<i>CONX-NACK</i>	<i>ERR-IND</i> <i>IDLE</i>	<i>IDLE</i>	<i>ERR-IND</i> <i>ATT-START</i>	<i>ERR-IND</i> <i>CNX</i>	<i>ERR-IND</i> <i>ATT-ACK</i>	<i>ATT-DCNX</i>
Data transmission request				<i>DATA-MSG</i> <i>CNX</i>	<i>DATA-MSG</i> <i>ATT-ACK</i>	
<i>ERR-IND</i>	<i>IDLE</i>	<i>IDLE</i>	<i>DCNX-REQ</i> <i>ATT-DCNX</i>	<i>DCNX-REQ</i> <i>ATT-DCNX</i>	<i>DCNX-REQ</i> <i>ATT-DCNX</i>	<i>ATT-DCNX</i>
Acknowledgement request				<i>SYNC-REQ</i> <i>ATT-ACK</i>		
<i>SYNC-ACK</i>	<i>ERR-IND</i> <i>IDLE</i>	<i>ERR-IND</i> <i>IDLE</i>	<i>ATT-START</i>	<i>ERR-IND</i> <i>CNX</i>	<i>CNX</i>	<i>ATT-DCNX</i>
<i>START-REQ</i>	<i>ERR-IND</i> <i>IDLE</i>	<i>ERR-IND</i> <i>IDLE</i>	<i>START-ACK</i> <i>CNX</i>	<i>START-ACK</i> <i>CNX</i>	<i>START-ACK</i> <i>ATT-ACK</i>	<i>ATT-DCNX</i>
Disconnection request				<i>DCNX-REQ</i> <i>ATT-DCNX</i>		



## A.2 Client MMTP OUT Path

	<i>IDLE</i>	<i>ATT-CNX</i>	<i>CNX</i>	<i>ATT-ACK</i> (START-ACK)	<i>ATT-DCNX</i>
Connection request	CONX-REQ <i>ATT-CNX</i>				
CONX-ACK	ERR-IND <i>IDLE</i>	START-REQ <i>ATT-ACK</i>	ERR-IND <i>CNX</i>	ERR-IND <i>ATT-ACK</i>	<i>ATT-DCNX</i>
CONX-NACK	ERR-IND <i>IDLE</i>	<i>IDLE</i>	ERR-IND <i>CNX</i>	ERR-IND <i>ATT-ACK</i>	<i>ATT-DCNX</i>
ERR-IND	<i>IDLE</i>	<i>IDLE</i>	DCNX-REQ <i>CNX</i>	DCNX-REQ <i>ATT-ACK</i>	<i>ATT-DCNX</i>
DATA-MSG	ERR-IND <i>IDLE</i>	ERR-IND <i>IDLE</i>	process the data <i>CNX</i>	ignore the data <i>ATT-ACK</i>	<i>ATT-DCNX</i>
SYNC-REQ	ERR-IND <i>IDLE</i>	ERR-IND <i>IDLE</i>	SYNC-ACK <i>CNX</i>	SYNC-ACK <i>ATT-ACK</i>	<i>ATT-DCNX</i>
Restart request			START-REQ <i>ATT-ACK</i>		
START-ACK	ERR-IND <i>IDLE</i>	ERR-IND <i>IDLE</i>	ERR-IND <i>CNX</i>	<i>CNX</i>	<i>ATT-DCNX</i>
Disconnection request			DCNX-REQ <i>ATT-DCNX</i>		
DCNX-REQ	ERR-IND <i>IDLE</i>	ERR-IND <i>IDLE</i>	DCNX-ACK <i>IDLE</i>	DCNX-ACK <i>IDLE</i>	DCNX-ACK <i>IDLE</i>
DCNX-ACK	ERR-IND <i>IDLE</i>	ERR-IND <i>IDLE</i>	ERR-IND <i>CNX</i>	ERR-IND <i>ATT-ACK</i>	<i>IDLE</i>

## ***B. Special Administrative Data Formats***

### **B.1 A1 Admin Data Type**

Field name	Offset	Length	Type	Value	Way	Tech. Field	Comment
Type	0	2	char	A1			Standard field - Refer to Section 5.7.3
MsgId	2	24	char				Standard field - Refer to Section 5.7.3
SendTime	26	12	digits				Standard field - Refer to Section 5.7.3
ReceiptTime	38	12	digits				Standard field - Refer to Section 5.7.3
DeliveryTimeout	50	6	digits				<ul style="list-style-type: none"> <li>• Format: HHmmss</li> <li>• This field is a timestamp, NOT a delay</li> <li>• <b>For future use</b></li> </ul>
ReceiveTime	56	12	digits		Out		The date and time of reception assigned by the access point
SubsId	68	11				<b>D_SUBS</b>	<p>MMTP OUT path: ID of the HUB subscriber that sent the message</p> <p>MMTP IN path: allows the sender to specify a HUB subscriber ID. This field may be used for comparisons with the D_SUBS technical field.</p>

Field name	Offset	Length	Type	Value	Way	Tech. Field	Comment
MemberId	79	8				<b>D_MBR</b>	MMTP OUT path: ID of the member that sent the message MMTP IN path: allows the sender to specify a member ID. This field may be used for comparisons with the D_MBR technical field.
Domain	87	1				<b>DOMAIN</b>	MMTP OUT path: specifies to what type of application the message should be distributed: P (Public), M (Member), S (Subscriber) MMTP IN path: allows the sender to specify a HUB subscriber ID. This field may be used for comparisons with the DOMAIN technical field.
Dest	88	11	char				<ul style="list-style-type: none"> <li>• Technical fields: <ul style="list-style-type: none"> <li>➤ for the <b>MMTP OUT</b> path: Not meaningful</li> <li>➤ for the <b>MMTP IN</b> path: Internal use only</li> </ul> </li> </ul>
Filler	99	29					Characters to fill A1 administrative data to its defined size (see Note below).

**Note:** Size of A1 administrative data: 128

## B.2 G0 Admin Data Type

Field name	Offset	Length	Type	Value	Way	Tech. Field	Comment
Type	0	2	char	G0			The admin data type. G0 is the "gateway" data type used only for communication between two HUBs.
MsgId	2	24	char				<ul style="list-style-type: none"> <li>Message identifier assigned by the sender.</li> <li><b>msgid is UNIQUE for each HUB subscriber and for each message sent</b></li> </ul>
SendTime	26	12	digits				<ul style="list-style-type: none"> <li>Send timestamp</li> <li>Format: MMDDHHmssc</li> <li>Date and time of transmission, assigned by the MMTP client when the message is sent to the CAP.</li> </ul>
ReceiptTime	38	12	digits				<ul style="list-style-type: none"> <li>Format: MMDDHHmssc</li> <li>The date and time of transmission assigned by the CAP when sending a message to the MMTP client.</li> </ul>
OnSubscriber	50	11	char				Mnemo of the HUB subscriber that sent the message
OnMember	61	8	char				Mnemo of the HUB member that sent the message
Domain	69	1	char				Specifies to what type of application the message is to be sent: P (Public), M (Member), S (Subscriber)
ClassId	70	4	char				ID of message class
Destination	74	11	char				Destination subscriber or member name for Subscriber or Member classes
nb_hop	85	1	digit				Time to live???
ReceiveTime	56	12	digits		Out		The date and time of reception assigned by the access point
Filler	86	29					Characters to fill G0 administrative data to its defined size (see Note below).
Ett	126	128					Original admin data

**Note:** Size of G0 administrative data: 255