CCS A.11

Q/SZSE

深圳证券交易所企业标准

Q/SZSE 0001—2024

组播行情数据分发协议规范

Specification of Market Data Distribution Protocol

2024-11-19 发布　　2024-11-19 实施

深圳证券交易所　　发布

目　　次

前　言

本文件按照GB/T　1.1—2020《标准化工作导则　第1部分：标准化文件的结构和起草规范》的规定起草。

本文件由深圳证券交易所提出。

本文件由深圳证券交易所归口管理。

本文件起草单位：深圳证券交易所、深圳证券通信有限公司、深圳证券信息有限公司。

本文件主要起草人：喻华丽、万春波、谢文海、郭未、王海、王冰、陈明忠、杨建明、王鹏飞、林林、谭笑、曹兆勇、谭浩通、詹宇彬、张剑。

组播行情数据分发协议规范

1 范围

本文件规定了深圳证券交易所与各市场参与主体之间使用组播协议分发实时行情的协议接口（Market Data Distribution Protocol，简称MDDP），包括类型定义、报文结构、报文定义。

本文件适用于深圳证券交易所实时组播行情数据分发、接收时的接口定义和软件开发。

2 规范性引用文件

本文件没有规范性引用文件

3 术语和定义

下列术语和定义适用于本文件。

3.1

行情 market data

交易所交易系统向市场发布的证券或合约价格信息。

[来源：JR/T 0292—2023，3.8]

3.2

行情快照 market data snapshot

在交易系统内部或是向市场发送的某一时刻所有合约的行情信息。

[来源：JR/T 0145—2016，2.14]

3.3

交易所 exchange

供已发行的证券进行流通转让或者期货等衍生品合约买卖的场所。

[来源：JR/T 0145—2016，2.1；JR/T 0292—2023，3.1]

3.4

交易所端 exchange side

指本协议连接的两端中，属于交易所系统的一端。

[来源：JR/T 0016—2014，3.2.8]

3.5

频道 channel

行情数据有序发送的最小单位。同一频道内的数据内容有序。

3.6

行情源 data feed

交易所端向市场发布行情数据的系统或服务。

3.7

组播流 multicast stream

行情源在一个组播地址和端口上发送的实时行情数据。一个组播流可以发送多个频道的行情数据。

3.8

数据流 data stream

行情源在一个组播流上发布的单个频道的实时行情数据。

3.9

数据包 data packet

数据流上发布的一个完整的 MDDP 报文。

3.10

发送源标识 sender id

交易所端行情源发送标识。通过该标识可识别行情数据来自哪一个发送者，也可以识别同一个行情源是否发生重启。

4 数据类型

本文件使用的所有整数类型默认均采用网络字节序（BIG-ENDIAN ）进行传输，通信双方可根据需要自行约定字节序。本文件支持的数据类型见表1。

表 1 数据类型

类型	描述
Int8	8位有符号整数
Int16	16位有符号整数
Int32	32位有符号整数
Int64	64位有符号整数
uInt8	8位无符号整数
uInt16	16位无符号整数
uInt32	32位无符号整数
uInt64	64位无符号整数
Char[x]	Char[x]表示字符串，x表示字符串最大字节数，字符串实际长度小于字段类型最大长度时右补空格；字符串默认使用UTF-8编码，通信双方可根据需要自行约定编码格式。
VarChar	变长字符串，长度由前导的VarCharLength指定；字符串默认使用UTF-8编码，通信双方可根据需要自行约定编码格式。
VarCharLength	变长字符串长度，uInt32
RawData	变长二进制数据，长度由前导的RawDataLength指定
RawDataLength	变长二进制数据长度，uInt32
Padding[x]	无格式数据，填充位

5 报文结构

5.1 通用结构

本文件报文由报文头、报文体、报文尾三部分构成，报文结构见图 1。

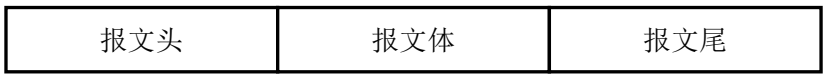


图 1　报文结构

5.2　报文头

报文头不加密传输，报文头结构见表 2。

表 2　报文头

域名	类型	长度（byte）	意义
Protocol	uInt8	1	协议标识，固定为0xFF
Version	uInt8	1	版本号，当前为0x01
HeaderSize	uInt8	1	整个报文头的长度，指示报文头由几个4Byte words组成。
SenderId	uInt8	1	发送源标识
MarketId	uInt16	2	市场标识
Channel	uInt16	2	频道号，同一市场内的频道不重复，同一频道内的数据内容有序。
SeqNum	Int64	8	报文序列号
MsgCount	uInt16	2	报文内消息条目数
Flag	uInt16	2	报文标志位
TotalFragments	uInt16	2	分片总数（可选）
FragmentNo	uInt16	2	分片序号（可选）
EncodeChecksum	uInt32	4	编码校验和（可选），编码前数据的校验和
Flagx	uInt16	2	扩展标志位x（可选），x为第N次扩展
Padding	Byte[x]	x	字节对齐填充位（可选），确保报文头长度为4字节整数倍，无意义。

——HeaderSize 整个报文头的长度，长度计算从报文头的第一个字段 Protocol 开始到最后一个字段 Padding 结束(如 HeaderSize = 5 表示报文头总长度为 5 * 4Byte = 20Bytes)。

——SenderId 用于识别同一个组播地址及端口上的不同发送者。

——MarketId 预留字段，目前固定填 1。

——Channel 行情数据可根据业务（如按证券集）分为不同的类别，每个类别的行情数据则可根据其数据量大小进一步分为多个频道。

——SeqNum 每个发送者会在对应市场的 Channel 组合内维持一个从 1 开始递增的序列号，报文内的每一条消息都会让序列号递增 1，下一个报文的 SeqNum 为上一个报文最后一笔消息的 SeqNum + 1。根据 ResendBySeqNum 标志位的不同，SeqNum 有两种编码机制：

- ResendBySeqNum = 0，发送者启动或重启后的首个报文从 1 开始编号。
- ResendBySeqNum = 1，发送者使用报文内第一条消息的序列号作为报文序号。（应用消息存在序号并连续递增时才可以使用该模式，如逐笔行情）

——MsgCount 该字段指示报文体中消息条目数。

——Flag 用于标识报文属性，左起第一位为 Bit15，各标志位含义见表 3。

表 3　标志位

Flag Bit	意义
Bit 15	PossDupFlag标志位，可能重复标志 标识报文中的数据为可能重复的数据，此时SeqNum可以小于期望的序列号。数据接收者如果已经接收到该数据则可以直接丢弃。 0：非可能重复报文 1：可能重复的报文
Bit 14-13	PacketType标志位，报文类型标志 00：管理报文 01：应用报文
Bit 12	ResendBySeqNum标志位 是否支持按报文的SeqNum进行重传 0:不支持 1:支持
Bit 11-10	Compress标志位，压缩标志 00:不压缩 01:zlib压缩 10:预留 11:预留
Bit 9-8	Encryption标志位，加密标志 00:不加密 01:自定义加密 10:预留 11:预留
Bit 7	MsgHeader标志位,前导长度标志位 0：前导长度不存在 1：前导长度存在
Bit 6	Fragment标志位，分片包标志 0:不支持分片包； 1:支持分片，将MDDP数据包按照指定包大小分为多个MDDP分片包发布
Bit 5	编码校验和标志位，0:不提供校验和；1:提供校验和
Bit 4-1	预留
Bit 0	Flag扩展位 0:下一个Flag扩展位不存在； 1:下一个Flag扩展位存在，报文头中会出现下一个扩展Flag标志位Flagx

——TotalFragments 可选字段。表示同一个 MDDP 数据包被分为几个分片。当支持分片时，报文头包含本字段。

——FragmentNo 可选字段。从 1 开始计数，表示同一个 MDDP 数据包的第几个分片。当支持分片时，报文头包含本字段。

——EncodeChecksum 可选字段。校验和算法为 Adler32。当编码校验和标志位设置时，报文头包含本字段。可通过该字段对解码后的报文体内容进行校验，如果校验和与本字段不一致则说明解码异常。

——Flagx 可选字段,扩展标志位 x,x 可以是 1,2,3...。可通过 Flag 标志位字段的扩展标志位 Bit0 新增一个扩展 Flag 标志位字段 Flagx。Flagx 的扩展标志位 Bit0 同样可以继续扩展另外一组 Flag 标志位 Flagx+1。多个扩展标志位及对应可选字段的排列顺序见表 4：

表 4　扩展标志位及对应可选字段的排列顺序

域名	类型	长度
Flag	uInt16	2
OptionField1	xxx	xx
OptionField2	xxx	xx
Flag1	uInt16	2
OptionField3	xxx	xx
OptionField4	xxx	xx

——Padding 可选字段，报文头长度不为 4Byte 的整数倍时，包头包含这个字段。

5.3　报文体

报文体由MDDP消息头和MDDP消息体两部分构成，MDDP消息头指示了报文中每一条应用消息的长度，MDDP消息体存储了具体的应用消息。MDDP消息头为可选项，由报文头中的MsgHeader标志位标识，带消息头的报文体结构见表5。

表 5　带 MDDP 消息头的报文体

域名	类型	长度（byte）	意义
Lengths	uInt32[x]	4 * MsgCount	MDDP消息头，指示了报文中每一条消息的长度
Body	Byte[x]	x由Lengths指定	MDDP消息体

5.4　报文尾

报文尾定义了校验和，不加密传输。校验和的计算范围从报文头开始（包括报文头）一直到报文体结束。校验和算法为Adler32。当支持分片时，每个分片独立计算校验和。报文尾结构见表6。

表 6　报文尾

域名	类型	长度（byte）	意义
Checksum	uInt32	4	校验和

6　报文定义

6.1　组播心跳报文

组播流由组播地址、端口及SenderId唯一标识，当组播流上没有任何应用层数据时，发送者会在组播流上周期性的发送组播心跳报文，帮助数据接收者检测组播通道状态。组播心跳报文的发送间隔为5秒，数据接收者在3个心跳周期内没有收到任何数据时，说明该路组播源故障。

Channel为0时表示该报文为“组播心跳报文”，组播心跳报文没有报文体，收到后可以直接丢弃。组播心跳报文结构见表7。

表 7　组播心跳报文

域名	类型	取值
Protocol	uInt8	协议标识，固定为0xFF
Version	uInt8	版本号，当前为0x01
HeaderSize	uInt8	5
SenderId	uInt8	发送源标识
MarketId	uInt16	1
Channel	uInt16	固定为0
SeqNum	Int64	0
MsgCount	uInt16	0
Flag	uInt16	高位→低位 0000000000000000
CheckSum	uInt32	校验和

6.2　数据流心跳报文

数据流由Channel及SenderId唯一标识，当报文的ResendBySeqNum标志位为1时，发送者会在数据流上周期性的发布数据流心跳报文，用于告知数据接收者当前数据流上已发送的最后一笔数据消息对应的序号。

本文件将“数据流心跳报文”的SeqNum设置为当前数据流上已发送的最后一笔消息的SeqNum。当数据流上没有发送过任何应用消息时，“数据流心跳报文”的SeqNum为0。

Channel不为0且MsgCount 为0时表示该报文为“数据流心跳报文”，数据流心跳报文没有报文体。数据流心跳报文结构见表8。

表 8　数据流心跳报文

域名	类型	取值
Protocol	uInt8	协议标识，固定为0xFF
Version	uInt8	版本号，当前为0x01
HeaderSize	uInt8	5
SenderId	uInt8	发送源标识
MarketId	uInt16	1
Channel	uInt16	2011
SeqNum	Int64	9999
MsgCount	uInt16	0
Flag	uInt16	高位→低位 0000000000000000

CheckSum	uInt32	校验和

6.3 数据流结束报文

“数据流结束报文”为特殊的数据流心跳报文，用于告知数据接收者当前数据流上所有业务数据已经发送完毕。

Channel 不为 0 且 MsgCount 为 0xFFFF（或写作 65535）时表示该报文为“数据流结束报文”。数据流结束报文结构见表 9。

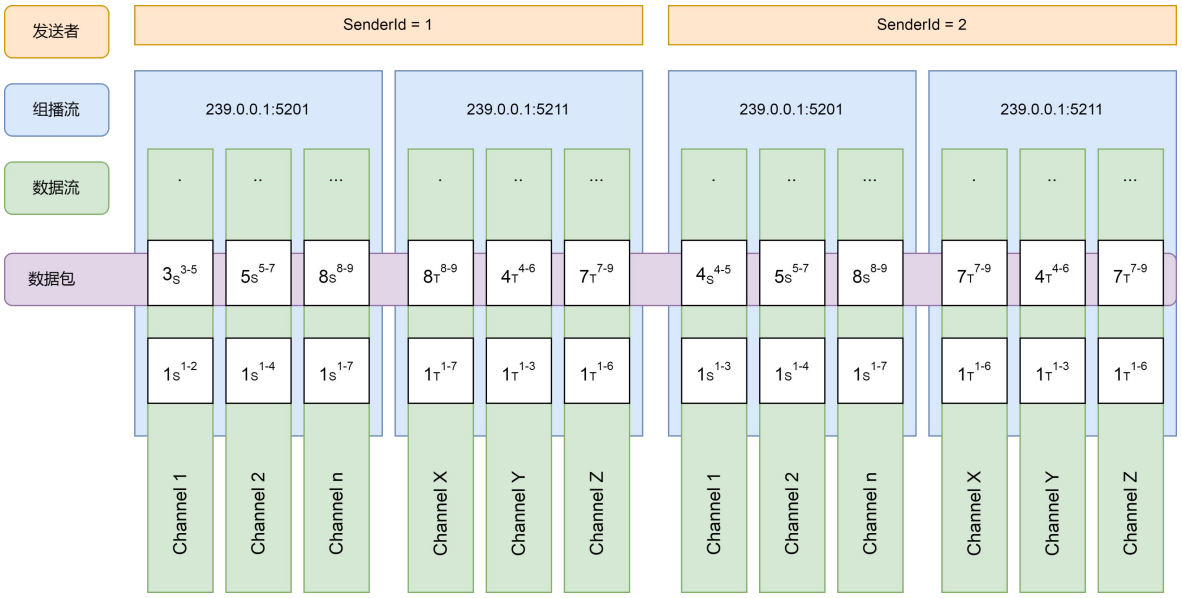
表 9 数据流结束报文

域名	类型	取值
Protocol	uInt8	协议标识，固定为0xFF
Version	uInt8	版本号，当前为0x01
HeaderSize	uInt8	5
SenderId	uInt8	发送源标识
MarketId	uInt16	1
Channel	uInt16	2011
SeqNum	Int64	9999
MsgCount	uInt16	0xFFFF
Flag	uInt16	高位→低位 0000000000000000
CheckSum	uInt32	校验和

附 录 A
（资料性）
数据流及编解码机制描述

A.1 发送者-组播流-数据流-数据包关系

本文件所描述发送者、组播流、数据流、数据包之间的关系见图A.1



发送者——指交易所端行情源，主备行情源通过 SenderId 标识，互为备份

数据包——数据流上发布的完整报文

[a] NSx-y,N 表示包序号,S 表示快照数据流,x-y 本数据流上的第 x 到第 y 笔快照消息，ResendBySeqNum 标志位为 0

[b] NTx-y,N 表示包序号,T 表示逐笔数据流,x-y 本数据流上的第 x 到第 y 笔逐笔消息，ResendBySeqNum 标志位为 1

图 A.1 发送者-组播流-数据流-数据包关系图

A.2 SenderId编码机制

SenderId 编码机制如下：

a) 交易日首次启动时，SenderId 设置为 Sender 在集群中的索引

b) 每次重启，SenderId = Sender 在集群中的索引 + （ 启动次数-1）* 集群的 Sender 数量

c) 当 SenderId > 255 时，SenderId 重置为 Sender 在集群中的索引

SenderId 编码机制示例见表 A.1。

表 A.1　SenderId 编码机制示意

集群	发送者索引	第1次启动	第2次启动	第N次启动
Sender1	0	0	2	0+(N-1)*2
Sender2	1	1	3	1+(N-1)*2

注 1：如果交易日内重启次数过多或是硬件故障导致重启后SenderId无法从磁盘恢复,此时SenderId重置为Sender在集群中的索引

A.3　SeqNum连续性检测机制

数据流上每个数据包的报文序号连续递增，由于组播会发生乱序、丢包和重包，因此数据接收者应该对数据流上的数据包报文序号进行检查。

a) SeqNum < 下一个报文期望的报文序号 NextExpectedSeqNum，此时认为报文为过期数据直接丢弃。

b) SeqNum = NextExpectedSeqNum，则处理报文，并设置下一个报文期望的报文序号 NextExpectedSeqNum = SeqNum + MsgCount

c) SeqNum > NextExpectedSeqNum，说明可能发生了乱序或丢包，可采用两种处理机制:

1) 直接判定为网络丢包，从当前数据包开始继续处理后续报文。网络基础设施稳定，很少出现乱序时，可以考虑使用该机制。

2) 先判定网络发生乱序并缓存数据,当缓存的数据超过预设的缓冲区大小或超时后依然未收到预期的报文，则判定为网络丢包。使用缓存中序号最小报文的作为 NextExpectedSeqNum，继续处理缓存中以及后续的报文。

注 2：接收者的乱序缓冲区需根据网络乱序情况进行设置，如果网络基础设施稳定，很少出现乱序时，则可以设置为一个比较小的值（如:16）

注 3：可重传的数据发生丢包时的重传机制见附录 A.9 恢复机制章节

A.4　行情源故障检测机制

行情源故障重启后，不支持按照 SeqNum 重传类报文(ResendBySeqNum 标志位为 0)的 SeqNum 会从 1 开始重新编号，从而导致报文的 SeqNum 回退。本文件提供两种故障检测机制，便于接收者对行情源故障进行识别并进行恢复：

——SenderId 发生变更

当接收者发现数据流上的 SenderId 发生变更时，说明行情源发生重启或是发生了交易日切换，此时应使用最新的数据报文中的消息序号重置下一个期望的组播报文序号。

——SenderId 未发生变更，但 SeqNum 回退

该场景一般是因为硬件故障导致 SenderId 无法恢复，从而导致重启后 SenderId 未发生变化但 SeqNum 发生了回退。由于组播可能发生乱序，因此接收者可配置一个阈值，当 SeqNum + 阈值 < NextExpectedSeqNum 则认为是行情源重启，此时应使用最新的数据报文中的消息序号中重置下一个期望的组播报文序号。否则，则将报文认为是乱序包进行处理。

A.5 消息解码机制

本文件当前支持分发深圳证券交易所BINARY行情数据，MDDP消息体内存储的应用消息定义见《深圳证券交易所Binary行情数据接口规范》。MDDP消息头提供了MDDP消息体内每一条消息的长度，当MDDP消息头存在时，接收者可以根据MDDP消息头提供的消息长度信息，在不解析MDDP消息体的情况下，实现消息的切割。带MDDP消息头的报文体结构见图A.2。

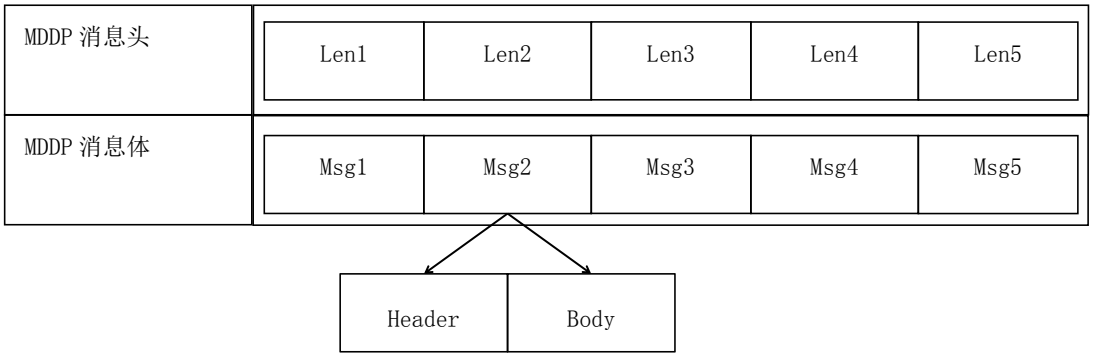


图 A.2 带消息头的报文体结构

应用层消息的Header及Body定义按照《深圳证券交易所Binary行情数据接口规范》第四章描述的消息定义进行处理。MDDP消息头使用示例见表A.2。

表 A.2 MDDP 消息头使用示例

消息	消息偏移量
Msg1	Offset1 = 0
Msg2	Offset2 = Offset1 + Len1
Msg3	Offset3 = Offset2 + Len2
Msg4	Offset4 = Offset3 + Len3
Msg5	Offset5 = Offset4 + Len4

注 4：本文件取消了《深圳证券交易所 Binary 行情数据接口规范》4.2.2 中每条消息对应的校验和，通过本文件 5.4 章节定义的报文尾中的校验和对整个报文中的所有内容进行完整性校验。接收者对 MDDP 报文进行完整性校验后，直接通过 MDDP 消息头完成消息切割并解码消息即可，无需对每一条消息再进行单独的完整性校验。

A.6 打包机制

为了提高发送效率，MDDP会将多条消息打包至一个MDDP数据包中发送。MDDP发送端按照合并、压缩、加密及分片的顺序对发送数据做打包处理，MDDP接收端按照组合、解密、解压及拆分的顺序对接收数据做解包处理。

a) 打包流程见图 A.3

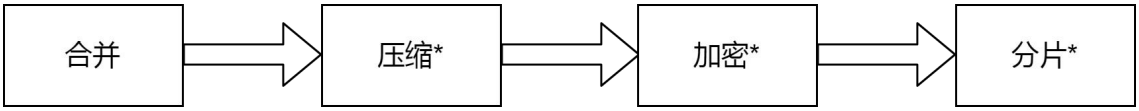


图 A.3　打包流程

1) 将多条消息组装到一个 MDDP 数据包中。
2) 使用 adler32 算法计算原始数据包校验和。
3) 如果启用压缩机制，则对原始数据进行压缩处理。
4) 如果启用加密机制，则对上一步骤产生的数据进行加密处理。
5) 如果启用分片机制，则根据分包机制对上一步骤产生的数据进行分包处理。
6) 将原始数据包校验和加入报文头扩展位中，并设置编码校验和标志位。

b) 解包流程见图 A.4

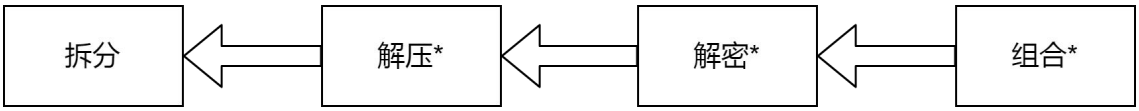


图 A.4　解包流程

1) 如果启动分片机制，则根据分包机制将多个 MDDP 分片合并一个 MDDP 数据包。
2) 如果启用加密机制，则对上一步骤产生的数据进行解密处理。
3) 如果启用压缩机制，则对原始数据进行解压处理。
4) 对上一步中的数据用 adler32 算法计算校验和与报文头中的编码校验和比较，验证数据的正确性。
5) 根据 MDDP 报文头或是应用层协议将数据包中的内容拆分成消息

A.7　分包机制

本文件支持按照指定大小将MDDP数据包切分为多个MDDP分片包。启用分片机制后，MDDP发送方会按照指定的组播包大小对MDDP数据包进行分片处理，分片后每一个分片包大小（包含报文头+报文体+报文尾）不超过指定组播包大小，每一个分片包通过一个组播包进行发送。

启用分片机制后一条消息可能会跨多个分片传输，接收者应将多个分片包组合为一个数据包后再进行解码操作。如果启用了压缩或是加密机制，则同样应将MDDP分片包合并为一个MDDP数据包后再进行解密和解压操作。分包、组包流程见图A.5。

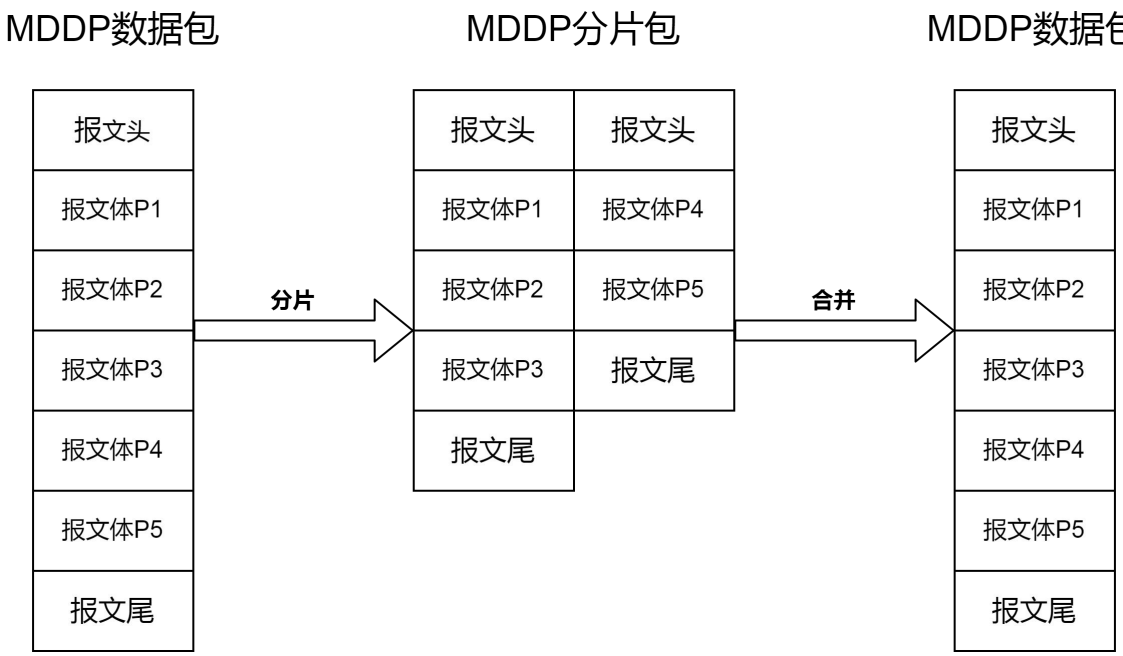


图 A.5　分包组包流程

无压缩加密场景下，分包机制示例如下：

a)　MDDP 数据包报文长度超过指定组播包大小，数据包内容见表 A.3，报文体中的数据仅用于示例展示，无实际意义。

表 A.3　超过指定组播包大小报文示例

域名	类型	取值
Protocol	uInt8	协议标识，固定为0xFF
Version	uInt8	版本号，当前为0x01
HeaderSize	uInt8	5
SenderId	uInt8	发送源标识
MarketId	uInt16	1
Channel	uInt16	2011
SeqNum	Int64	1
MsgCount	uInt16	10
Flag	uInt16	高位→低位:xxxxxxxxxx0xxxxx
TotalFragments	uInt16	0
FragmentNo	uInt16	0
报文体	Char[x]	AAAAAAAABBBBBBBBBCCC
CheckSum	uInt32	...

b) 发送者根据指定组播包大小将一个 MDDP 数据包分成多个 MDDP 分片包，并为每个 MDDP 分片包生成校验和。MDDP 分片包为 Fragment 标志位 1 的数据包，分片包内容见表 A.4、表 A.5、表 A.6。

表 A.4 分片包 1 示例

域名	类型	取值
Protocol	uInt8	协议标识，固定为0xFF
Version	uInt8	版本号，当前为0x01
HeaderSize	uInt8	5
SenderId	uInt8	发送源标识
MarketId	uInt16	1
Channel	uInt16	2011
SeqNum	Int64	1
MsgCount	uInt16	10
Flag	uInt16	高位→低位:xxxxxxxxxx1xxxxx
TotalFragments	uInt16	3
FragmentNo	uInt16	1
报文体	Char[x]	AAAAAAAA
CheckSum	uInt32	...

表 A.5 分片包 2 示例

域名	类型	取值
Protocol	uInt8	协议标识，固定为0xFF
Version	uInt8	版本号，当前为0x01
HeaderSize	uInt8	5
SenderId	uInt8	发送源标识
MarketId	uInt16	1
Channel	uInt16	2011
SeqNum	Int64	1
MsgCount	uInt16	10
Flag	uInt16	高位→低位:xxxxxxxxxx1xxxxx
TotalFragments	uInt16	3
FragmentNo	uInt16	2
报文体	Char[x]	BBBBBBBBB
CheckSum	uInt32	...

表 A.6 分片包 3 示例

域名	类型	取值
Protocol	uInt8	协议标识，固定为0xFF
Version	uInt8	版本号，当前为0x01
HeaderSize	uInt8	5
SenderId	uInt8	发送源标识

MarketId	uInt16	1
Channel	uInt16	2011
SeqNum	Int64	1
MsgCount	uInt16	10
Flag	uInt16	高位→低位:xxxxxxxxxx1xxxxx
TotalFragments	uInt16	3
FragmentNo	uInt16	3
报文体	Char[x]	CCC
CheckSum	uInt32	...

c) 接收者收到 MDDP 分片包后，对 MDDP 分片包的校验和进行校验，校验通过后保存该分片包。当收齐一个 MDDP 数据包的所有分片包后，应将多个分片包按照顺序组成 MDDP 数据包，再进行解码操作。

A.8 加解密机制

a) 本文件支持对行情数据进行加密传输，具体机制如下：

1) 发送者将数据包 Encryption 标志位设置为"01:自定义加密"，并使用令牌对报文体进行异或操作后再对外发布数据
2) 接收者收到的数据包 Encryption 标志位如果为"01:自定义加密"，则先使用令牌对报文体进行异或操作，再进行其他解码操作
3) 对于启用了分包机制的分片包，接收者应先将分片包组合成数据包后再进行解码
4) 令牌获取机制见本文件附录 A 令牌获取机制章节

数据包启用自定义加密后的加解密流程见图A.6。

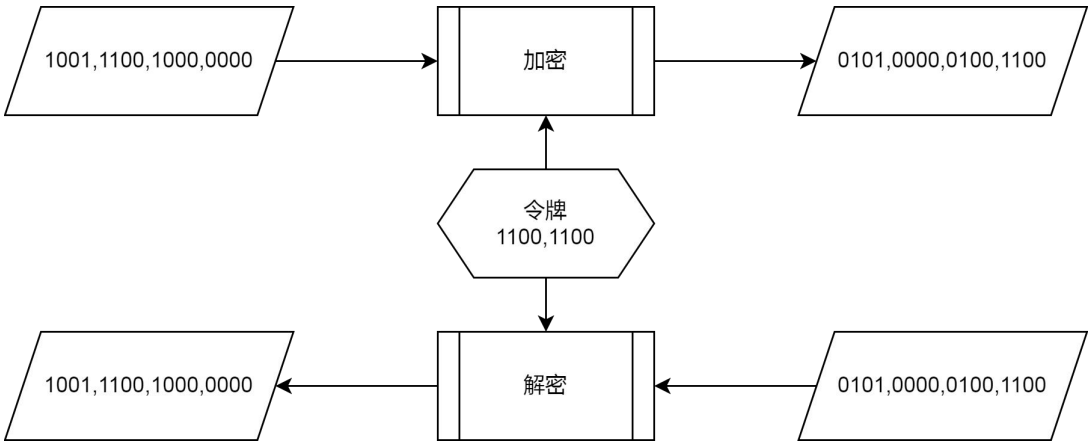


图 A.6 加解密流程

b) 对提供了原始数据校验和的数据包（EncodeChecksum 标志位为“1：提供校验和”),可通过报文头提供的 EncodeChecksum 字段校验解密后数据的正确性。

A.9 恢复机制

如果通过“SeqNum连续性检测机制”发现数据流出现丢包且丢包数据为可重传数据，则可以通过应用层提供的重传机制补传缺失数据，应用层提供的重传机制见《深圳证券交易所Binary行情数据接口规范》。

A.10 令牌获取机制

每个交易日，系统参与人应通过应用层提供的重传通道获取编解码令牌。因故障无法获取到令牌时，可将启用加密的数据包当做丢包处理。 应用层提供的令牌获取机制参见《深圳证券交易所Binary行情数据接口规范》。

参 考 文 献

[1] GB/T 1.1-2020 标准化工作导则 第1部分：标准化文件的结构和起草规则
[2] JR/T 0182—2020 轻量级实时STEP消息传输协议
[3] JR/T 0022—2020 证券交易数据交换协议
[4] JR/T 0103—2014 证券交易数据交换编解码协议
[5] 深圳证券交易所Binary行情数据接口规范