CHAPTER 26

TmNSDataMessage Transfer Protocol

LIST OF FIGURES

LIST OF TABLES

Distribution Statement A

Approved for public release: distribution unlimited.

26. CHAPTER 26

TmNSDataMessage Transfer Protocol

26.1 General

This chapter defines how *TmNS*-specific data (*TmNSDataMessages*) are transferred between applications. A *DataSource* shall transmit *TmNSDataMessages* and a *DataSink* shall receive *TmNSDataMessage*s. A *DataChannel* identifies a logical network connection used to transfer *TmNSDataMessages* between a *DataSource* and *DataSink*.

A unicast *DataChannel* is a logical network connection between a single *DataSource* and a single *DataSink*.

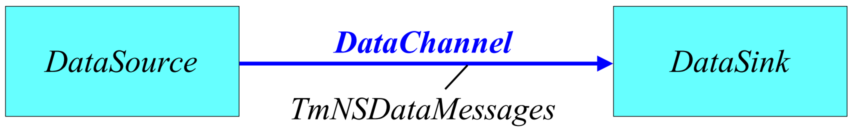

Figure 26-1. Unicast *DataChannel*

A multicast or broadcast *DataChannel* is a logical network connection between a single *DataSource* and one or more *DataSinks*.

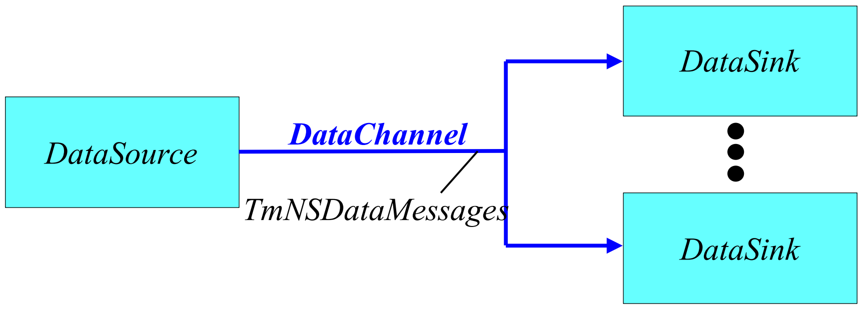

Figure 26-2. Multicast or Broadcast *DataChannel*

This document describes how *DataChannels* are allocated and managed via Application Data Transfer Resources. The associated management resources are defined in Chapter 25, Management Resources. The bit numbering, bit ordering, and byte ordering conventions used in this chapter are described in Appendix 21B.

A *DataChannel* may be established by submitting a *ResourceRequest* to specific Application Data Transfer Resources or via *Metadata* (i.e., described in an *MDL Instance Document*).

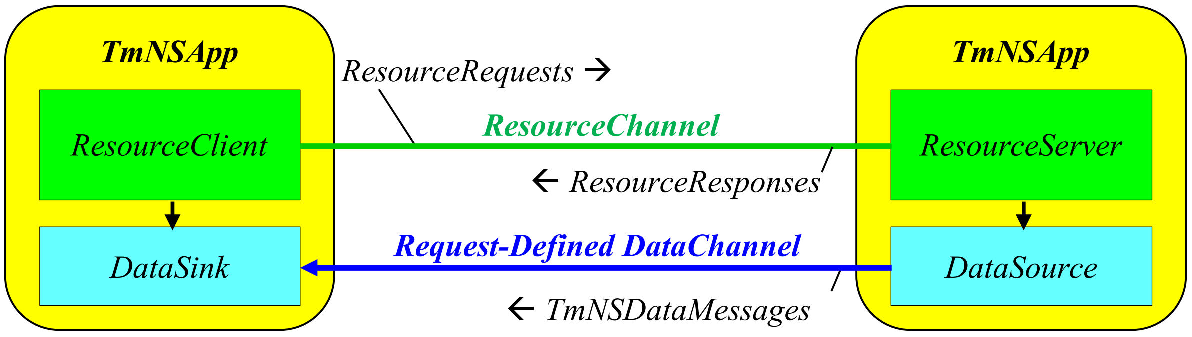

Figure 26-3. Request-Defined Data Channel

26.2 Data Channel Characteristics

The following information describes a *DataChannel*:

- Network Transport Characteristics
- Message List
- Time Range

26.2.1 Network Transport Characteristics

TmNSDataMessages shall be transported using either the User Data Protocol (UDP) or the Transport Control Protocol (TCP). A *DataChannel* shall support a single Differentiated Services Code Point (DSCP) assignment as specified in the Quality of Service section of Chapter 22, Network-Based Protocol Suite.

For a *Metadata*-Defined *DataChannel*, the Network Transport Characteristics are specified in an *MDL Instance Document*. See Section 26.3, *Metadata*-Defined Application Data Transfer, for more information.

For a Request-Defined *DataChannel*, the Network Transport Characteristics are included in the *ResourceRequest*. See Section 26.4, Request-Defined Application Data Transfer, for more information.

26.2.1.1 UDP DataChannel

UDP-capable DataSources

- shall support sending *TmNSDataMessages* via UDP/IP multicast, as specified in Chapter 22, Network-Based Protocol Suite.
- should support sending *TmNSDataMessages* via UDP/IP unicast or broadcast, as specified in Chapter 22, Network-Based Protocol Suite.
- shall send one complete *TmNSDataMessage* or *TmNSDataMessage* fragment per UDP datagram.

NOTE	It is anticipated that a future version of this chapter may allow for multiple *TmNSDataMessages* to be delivered in a single UDP datagram.

UDP-capable *DataSinks*

- shall support receiving *TmNSDataMessages* via UDP/IP multicast, as specified in Chapter 22, Network-Based Protocol Suite.
- should support receiving *TmNSDataMessages* via UDP/IP unicast or broadcast, as specified in Chapter 22, Network-Based Protocol Suite.

26.2.1.2 TCP DataChannel

TCP-capable *DataSources* shall support sending *TmNSDataMessages* via TCP/IP, as specified in Chapter 22, Network-Based Protocol Suite.

TCP-capable *DataSinks* shall support receiving *TmNSDataMessages* via TCP/IP, as specified in Chapter 22, Network-Based Protocol Suite.

26.2.2 Message List

A *TmNSDataMessage* List nominally contains a list of *MessageDefinitionIDs* and identifies which *TmNSDataMessages* shall be transported across the *DataChannel*.

For a Request-Defined *DataChannel*, the *TmNSDataMessage* list is included in the *ResourceRequest*.

For a *Metadata*-Defined *DataChannel*, the *TmNSDataMessage* list is defined in an *MDL Instance Document*. See Section 26.3, *Metadata*-Defined Application Data Transfer, for more information.

26.2.3 Time Range

A Time Range is comprised of a Start Time and End Time where each time specifies one of the following:

- Past Time: associated with retrieving data with timestamps before the current time
- Present Time: associated with current acquisition (e.g., live) data
- Future Time: associated with future acquisition data

For a Request-Defined *DataChannel*, the Time Range shall be included in the *ResourceRequest*. Time Ranges with various combinations of Past, Present, and Future Time are supported provided the End Time is greater than the Start Time.

26.3 *Metadata*-Defined Application Data Transfer

Metadata-Defined Application Data Transfer refers to the *TmNS*-specific application-level method of delivering *TmNSDataMessages* using an *MDL Instance Document* to specify *DataChannel* characteristics.

Metadata-Defined *DataChannels* are opened at *TmNSApp* startup / reconfiguration and remain open indefinitely.

26.3.1 Latency / Throughput Critical (LTC) Delivery Protocol

The LTC Delivery Protocol is the *TmNS*-specific application-level method of delivering *TmNSDataMessages* via UDP.

26.3.1.1 LTC Delivery Protocol Data Channel (*LTCDataChannel*)

LTCDataSources and *LTCDataSinks* shall support UDP Data Channels as defined in Section 26.2.1.1

LTCDataSources shall transport *TmNSDataMessages* using the UDP destination address and port determined by the following descending order of precedence:

1. The address and port associated with the *MDID* of the delivered *TmNSDataMessage* in the *MDL Instance Document.* If only the address is available, the default port is port 55555.
2. The broadcast IP address and port 55555.

LTCDataSources and *LTCDataSinks* shall comply with the standard *TmNSDataMessage* structure and mechanisms as specified in Chapter 24, Message Formats.

26.4 Request-Defined Application Data Transfer

Request-Defined Application Data Transfer refers to the *TmNS*-specific application-level method of delivering *TmNSDataMessages* via a *ResourceClient's* Data Request (*DataRequest*).

26.4.1 RTSP-based Control Channel (*RTSPControlChannel*)

DataSources and DataSinks (referred to as RTSPDataSources and RTSPDataSinks respectively) using the RTSPControlChannel shall exchange control commands and parameters using Real Time Streaming Protocol (RTSP), as specified in:

- RFC 2326: Real Time Streaming Protocol (RTSP)

RTSPDataSources and RTSPDataSinks shall transport RTSP commands in the RTSPControlChannel using TCP.

The *RTSPDataSink* shall act as the RTSP client and the *RTSPDataSource* shall act as the RTSP server.

The RTSP server shall listen for a TCP connection on the TCP port specified in the **ListeningPort** element under the **TmNSRCDataSource** element in the *MDL Instance Document*. If no port is specified, then port 55554 shall be used.

The RTSP client shall establish an *RTSPControlChannel* using the TCP port specified in the **ListeningPort** element under the selected **TmNSRCDataSource** element in the *MDL Instance Document*. If no port is specified, then port 55554 shall be used.

The *RTSPControlChannel* shall use the same DSCP in both directions based on the DSCP selected at origination of the *RTSPControlChannel* by the *RTSPDataSink*.

When an *RTSPDataSource* cannot perform in the manner specified in this standard, the *RTSPDataSource* shall issue the appropriate error Status-Code specified in RFC 2326.

An *RTSPDataSource* shall return all *TmNSDataMessages* that match a particular *TmNS_Request_Defined_URI* request and shall include an End of Data Indication (see Section 26.4.2.2).

The *RTSPControlChannel* shall support the following RTSP commands: "OPTIONS" "SETUP," "TEARDOWN," "PLAY," and "PAUSE" methods. Table 26-1 identifies the required RTSP Headers for the mandatory RTSP Methods.

Table 26-1 Required RTSP Header

Header	Type	Methods	Comment
Bandwidth	Request	PLAY	See Section 26.4.1.3 for details.
Connection	Request Response	ALL	Only applicable connection token is "close"
CSeq	Request Response	ALL	
Public	Response	OPTIONS	Only used in response to an OPTION request.
Range	Request Response	PLAY	See Section 26.4.1.2 for details.
Session	Request Response	PLAY, PAUSE, TEARDOWN	
Speed	Request	PLAY	See Section 26.4.1.3 for details.
Transport	Request Response	SETUP	See Section 26.4.1.1 for details.

RTSP clients and RTSP servers may support additional RTSP commands and associated header fields as specified in RFC 2326.

26.4.1.1 Transport Header

The RTSP "Transport" header shall be supported by *RTSPDataSources* and *RTSPDataSinks* using the *RTSPControlChannel*. The Transport Header indicates which transport protocol is to be used and configures its parameters such as destination address, multicast time-to-live and destination port. A Transport request header field may contain a list of transport options acceptable to the client. Transport options are comma separated, listed in order of preference. Parameters may be added to each transport option, separated by a semicolon. All *RTSPDataSources* and *RTSPDataSinks* shall support the following Transport Header parameters:

```
Transport            =    "Transport" ":"
                          1\#transport-spec
transport-spec       =    transport-protocol/profile[/lower-transport]
                          *parameter
transport-protocol   =    "TMNS"
profile              =    "TMNSP"
lower-transport      =    "TCP" | "UDP"
parameter            =    ( "unicast" | "multicast" )
                     |    ";" "destination" [ "=" address ]
                     |    ";" "ttl" "=" ttl
                     |    ";" "client_port" "=" port [ "-" port ]
ttl                  =    1*3(DIGIT)
port                 =    1*5(DIGIT)
address              =    host
```

NOTE	RFC 2326 states that a lower-transport protocol of "TCP" results in interleaving user-request data onto the *RTSPControlChannel*. This standard deviates from RFC 2326 by interpreting the lower-transport protocol of "TCP" as requiring a separate TCP data channel (not an interleaved control+data channel). See Section 26.4.2.

26.4.1.2 Range Header

The following PTP Time Range format shall be supported in the Range Header by *RTSPDataSources* and *RTSPDataSinks* using the *RTSPControlChannel*:

```
ptp-range     = "ptp-clock" "=" ptp-startTime "-" [ ptp-endTime ]
ptp-startTime = "start" | "now" | TmNStimestamp*
ptp-endTime   = "end"   | "now" | TmNStimestamp*
```

*`TmNStimestamp` format is defined in Chapter 22, Network-Based Protocol Suite in Section 22.5.1.3.6, TmNS Time Format.

The following rules shall be supported for the PTP time range:

- If a `ptp-endTime` is specified, then the `ptp-endTime` shall be greater than the `ptp-startTime`.
- A "`start`" constant shall be interpreted as the earliest **MessageTimestamp** of all available *TmNSDataMessages*.
  A "`now`" or "`end`" constant shall be interpreted as the latest **MessageTimestamp** of all available *TmNSDataMessages* at the receipt of the request.
- Not specifying a `ptp-endTime` or specifying a `ptp-endTime` that exceeds the latest **MessageTimestamp** of all available *TmNSDataMessages* results in the *RTSPDataSource* transmitting data from the `ptp-startTime` to the last available requested *TmNSDataMessage* and then continually transmitting received requested *TmNSDataMessages* until one of the following conditions occurs:

- The `ptp-endTime` is specified and the **MessageTimestamp** of a received requested *TmNSDataMessage* is equal to or exceeds the specified `ptp-endTime`.
- A TEARDOWN is executed.
- The TCP-based *RCDataChannel* is closed.
- For all of the above conditions except the *RCDataChannel* closure, the *RTSPDataSource* shall transmit an End of Data Indication (see Section 26.4.2.2) prior to closing the *RCDataChannel*.

• Requests with no `ptp-endTime` shall remain active until a TEARDOWN is executed.

• If no *TmNSDataMessages* are available, the *DataSource* shall return a Status-Code of 412 ("Precondition Failed") except in the case where the `ptp-startTime` is set to "`start`" or "`end`" and the `ptp-endTime` is not set.

• If a time range specification does not satisfy the aforementioned rules, the *RTSPDataSource* shall return a Status-Code of 457 ("Invalid Range").

To support *TmNSDataMessage's* native Message Timestamp format, *RTSPDataSources* and *RTSPDataSinks* implementing the *RTSPControlChannel* shall support the following modifications to RFC 2326, Section 12.29 ("Range"):

1. The PTP Time Range format shall be supported.
2. The NPT and UTC time range formats may be supported.
3. The "time=" option may be supported with the addition of the PTP time range format.

```
Range             = "Range" ":" 1\#ranges-specifier
                    [ ";" "time" "=" utc-time | TmNStimestamp[1] ]
ranges-specifier = npt-range | utc-range | smpte-range | ptp-range[1]
```

[1] Bold items are new; the remaining items are defined in RFC 2326.

If the RTSP "Range" header is not specified, then data shall be supplied as though a "`start`" constant was given for the `ptp-startTime`, an "`end`" constant was given for the `ptp-endTime` and no value for the "time" option was given.

The RTSP "Range" header represents only a request for a time range, and standard errors should be returned when requests cannot be serviced or in-progress connections fail.

NOTE	Use of Society of Motion Picture and Television Engineers (SMPTE) relative timestamps in the RTSP "Range" header is not recommended. SMPTE timestamps are intended for video clips and the format ("hours:minutes:seconds:frames.subframes") does not clearly map to time range selection based on *TmNSDataMessage* **MessageTimestamp** values.

NOTE	The inclusiveness and exclusiveness of range intervals is specified in Section 12.29 “Range” of RFC 2326.

RTSPDataSources and *RTSPDataSinks* shall interpret `ptp-startTime` and `ptp-endTime` values as measurement time, not as message time.

NOTE	The start and end time values must be interpreted as measurement time and not message time in order to ensure all requested data is returned. A message may contain data for a time range, not just a single time as specified by the message’s **MessageTimestamp**.

The first *TmNSDataMessage* returned for a specified `ptp-startTime` shall be the requested *TmNSDataMessage* with the latest **MessageTimestamp** that is less than or equal to the `ptp-startTime`.

If the `ptp-startTime` precedes the earliest available requested *TmNSDataMessage’s* **MessageTimestamp**, the earliest requested *TmNSDataMessage* shall be the first *TmNSDataMessage* returned.

26.4.1.3 Bandwidth and Speed Headers

The RTSP “Speed” header shall be supported by RTSPDataSources using the RTSPControlChannel.

The RTSP “Speed” header should be supported by RTSPDataSinks using the RTSPControlChannel.

For the RTSP “Speed” header, “normal” speed (1.0) shall be defined as the rate at which *TmNSDataMessage* MessageTimestamp values progress.

NOTE	Not all speeds from the RTSP “Speed” header are required to be supported.

The RTSP “Bandwidth” header shall be supported by *RTSPDataSources* using the *RTSPControlChannel*.

The RTSP “Bandwidth” header should be supported by *RTSPDataSinks* using the *RTSPControlChannel*.

If the RTSP "Speed" and "Bandwidth" headers are not specified, then data shall be supplied as fast as possible, as regulated by the resources between the *RTSPDataSource* and the *RTSPDataSink*.

RTSPDataSinks shall not specify both the RTSP "Speed" and "Bandwidth" headers in the same request.

26.4.1.4 Request-Defined URI Syntax

The RTSP methods used in the R*TSPControlChannel* shall use the *TmNS_Request_Defined_URI* to request specific data from an *RTSPDataSource*. The *TmNS_Request_Defined_URI* shall use the generic syntax for URIs as specified in Chapter 22, Network-Based Protocol Suite, as specialized below:

```
TmNS_Request_Defined_URI =
  "rtsp://" TmNShost [ ":" TmNShostport ] "/" "TmNS" "/" TmNSversion "/"
  [ TmNSlist "/" ] [ TmNSdestIP [ ":" TmNSdestport ] "/" ]
  [ "-" TmNSplaybackopt "/" ] [ "-" TmNStimeopt "/" ]
  [ "/" TmNSdeliveryDSCP ]

TmNShost          = TmNShostname | TmNSIPv4address
TmNShostname      = *( TmNSdomainlabel "." ) TmNStoplabel [ "." ]
TmNSdomainlabel   = TmNSalphanum | TmNSalphanum *( TmNSalphanum | "-" ) TmNSalphanum
TmNStoplabel      = ALPHA | ALPHA *( TmNSalphanum | "-" ) TmNSalphanum
TmNSIPv4address   = 1*DIGIT "." 1*DIGIT "." 1*DIGIT "." 1*DIGIT
TmNShostport      = 1*DIGIT
TmNSdeliveryDSCP  = 1*DIGIT

TmNSversion       = "1.0"

TmNSlist          = (1*TmNSmdidlist) | (1*(TmNSpdidlist ">" TmNSdeliverymdid)) |
                    (1*(TmNSmeasidlist ">" TmNSdeliverymdid "<" TmNSdeliverypdid ))

TmNSmdidlist      = 1*( "&" TmNSmdid [ "-" TmNSmdid ] )
TmNSmdid          = 1*DIGIT

TmNSpdidlist      = 1*( TmNSmdidlist 1*( "@" TmNSpdid [ "-" TmNSpdid ] ) )
TmNSpdid          = 1*DIGIT

TmNSmeasidlist    = 1*( TmNSpdidlist 1*( "#" TmNSmeasid ["-" TmNSmeasid ] ) )
TmNSmeasid        = 1*DIGIT

TmNSdestIP        = 1*DIGIT "." 1*DIGIT "." 1*DIGIT "." 1*DIGIT
TmNSdestport      = 1*DIGIT
TmNSdeliverymdid  = 1*DIGIT
TmNSdeliverypdid  = 1*DIGIT

TmNSplaybackopt   = "l" | "n" ; "l" = marked as live (acquired) data in MessageFlags
                            ; "p" = marked as playback data in MessageFlags
                            ; default is "a" if not provided

TmNStimeopt       = "o" | "c" ; "o" = original timestamps
                            ; "c" = timestamps based on RTSPDataSource current time
                            ; default is "o" if not provided

TmNSalphanum      = ALPHA | DIGIT
```

All numeric fields of the *TmNS_Request_Defined_URI* shall be interpreted as decimal.

The **TmNShost** and optional **TmNShostport** values shall indicate the IPv4 address and port of the *RTSPDataSource*.

The optional **TmNSdeliveryDSCP** specifies the DSCP marking to which requested *TmNSDataMessages* shall be sent. If the **TmNSdeliveryDSCP** is not specified, the *RTSPDataSource* shall mark all delivered IP packages with the "Best Effort" marking.

A **TmNSlist** that contains a **TmNSmdidlist** shall indicate a *MessageDefinitionID* request type according to Section 26.4.1.5.1. A request that does not include the **TmNSlist** shall indicate a *MessageDefinitionID* request for all *MessageDefinitionIDs*.

A **TmNSlist** that contains a **TmNSpdidlist** shall indicate a *PackageDefinitionID* request type according to Section 26.4.1.5.2.

A **TmNSlist** that contains a **TmNSmeasidlist** shall indicate a *MeasurementID* request type according to Section 26.4.1.5.3.

TmNSmdid values separated by a "-" shall indicate a request for an inclusive range of *MDIDs* between the first and last **TmNSmdid** values specified.

TmNSpdid values separated by a "-" shall indicate a request for an inclusive range of *PDIDs* between the first and last **TmNSpdid** values specified.

TmNSmeasid values separated by a "-" shall indicate a request for an inclusive range of *MeasurementIDs* between the first and last **TmNSmeasid** values specified.

For all request types:

- If present, the `TmNSdestIP` ":" `TmNSdestport` value shall indicate the IPv4 address and port to which requested *TmNSDataMessages* shall be sent.
- If present, the **TmNSplaybackopt** value indicates
  - The *PlaybackDataFlag* shall be set to 1'b0 when the value is "l"
  - The *PlaybackDataFlag* shall be set to 1'b1 when the value is "p"
  - If the **TmNSplaybackopt** is not present, the *PlaybackDataFlag* shall be set to 1'b0.
- If present, the **TmNStimeopt** value indicates
  - The *TmNSDataMessage* Message Timestamps shall be the original timestamp when the value is "o"
  - The *TmNSDataMessage* Message Timestamps shall be based on the *RTSPDataSource's* current time when the value is "c"
  - If the **TmNStimeopt** is not present, the *TmNSDataMessage* Message Timestamps shall be the original timestamp.
- When requesting *Packages* without standard *PackageHeaders* to be delivered using *Packages* with standard *PackageHeaders*, the time expressed using the delivery

MessageTimestamp and delivery *PackageTimeDelta* shall be equivalent to the time expressed by the requested **MessageTimestamp**.

NOTE	As noted in RFC 2068, “servers should be cautious about depending on URI lengths above 255 bytes because some older client or proxy implementations may not properly support these lengths.” The appropriate error `Status-Code` specified in RFC 2326 for `"Request-URI Too Large"` is `"414"`.

26.4.1.5 Request Types

RTSPDataSources shall return valid *TmNSDataMessages* based on the particular request type as described in the following sections.

If none of the requested *MessageDefinitionIDs* are defined in an *RCDataSource’s RCDataSource* list, the *RCDataSource* shall return a Status-Code of 412 (“Precondition Failed”).

If no *TmNSDataMessages* are available on the *RTSPDataSource* for all requested *MessageDefinitionIDs*, the *RTSPDataSource* shall transmit an End of Data Indication (see Section 26.4.2.2) prior to closing the *RCDataChannel*

NOTE	Since the *RTSPDataSource* returns ALL data that match its request criteria, it is possible that the combination of a particular request and data present at an *RTSPDataSource* will result in duplicate data being returned. The possibility of this data duplication can be reduced or eliminated by generating a more specific request.

26.4.1.5.1 MessageDefinitionID Request (TmNSmdid)

RTSPDataSources processing a *MessageDefinitionID* request shall adhere to the following requirements:

- All *TmNSDataMessages* matching the requested *MessageDefinitionID(s)* within the timeframe specified shall be delivered.
- Delivered *TmNSDataMessages* shall be labeled with the original *MessageDefinitionID(s)*.
- The delivered *TmNSDataMessages* Message Timestamp value is governed by the presence or absence of the **TmNStimeopt** value in the *TmNS_Request_Defined_URI.*
- Delivered *TmNSDataMessages* shall retain the **ApplicationDefinedFields***,* **MessageFlags***,* and **StatusFlags** fields from the original *TmNSDataMessages*.

26.4.1.5.2 PackageDefinitionID Request (TmNSpdid)

RTSPDataSources processing a *PackageDefinitionID* request shall adhere to the following requirements:

- Valid *TmNSDataMessages* shall be delivered containing the original *Packages* matching the requested *PackageDefinitionID(s)*. Instances of the *Packages* to be delivered may be refined through the specification of *MessageDefinitionIDs*;

otherwise, ALL instances of the *Packages* within the timeframe specified shall be delivered.

- Delivered *TmNSDataMessages* shall be labeled with the *MessageDefinitionID* set to the value specified in **TmNSdeliverymdid**.
- Delivered *TmNSDataMessages* shall follow the requirements in Section 26.5.4 for handling *MessageFlags* fields.
- Any *ApplicationDefinedFields* in the delivered *TmNSDataMessages* shall indicate conditions on the *RTSPDataSource* delivering the *TmNSDataMessages*, not the original *RTSPDataSource*.

26.4.1.5.2.1 ***PackageDefinitionID*** **Request Standard PackageHeader Handling**

RTSPDataSources processing a *PackageDefinitionID* request shall deliver all requested *Packages* from original *Packages* that use the standard *PackageHeader*.

26.4.1.5.2.2 PackageDefinitionID Request Non-Standard PackageHeader Handling

RTSPDataSources processing a *PackageDefinitionID* request and that support extraction from *Packages* that do not use the standard *PackageHeader* shall deliver all requested *Packages* from original *Packages* that do not use the standard *PackageHeader*.

26.4.1.5.3 MeasurementID Request (TmNSmeasid)

RTSPDataSources processing a *MeasurementID* request shall adhere to the following requirements:

- Valid *TmNSDataMessages* shall be delivered containing *Packages* with the *MeasurementData* matching the requested *MeasurementID(s)*. Instances of the *MeasurementData* to be delivered may be refined through the specification of *MessageDefinitionIDs* and/or *PackageDefinitionIDs*; otherwise, ALL instances of *MeasurementData* within the timeframe specified shall be delivered.
- Delivered *TmNSDataMessages* shall be labeled with the *MessageDefinitionID* field in the *TmNSDataMessageHeader* set to the value specified in **TmNSdeliverymdid**.
- Delivered *TmNSDataMessages* shall contain *Packages* according to the *PackageDefinition* corresponding to the **TmNSdeliverypdid**.
- Delivered *TmNSDataMessages* shall follow the requirements in Section 26.5.4 for handling **MessageFlags** fields.
- Any **ApplicationDefinedFields** in the delivered *TmNSDataMessages* shall indicate conditions on the *RTSPDataSource* delivering the *TmNSDataMessages*, not the original *RTSPDataSource.*
- Each requested *Package* containing the requested *MeasurementData* shall have a single corresponding delivery *Package*. A requested *Package* containing the requested *MeasurementData* shall not have more than one corresponding delivery *Package*.

26.4.1.5.3.1 MeasurementID Request Standard PackageHeader Handling

RTSPDataSources processing a *MeasurementID* request shall adhere to the following requirements:

- The *RTSPDataSource* shall deliver all requested *MeasurementData* from original *Packages* that use the standard *PackageHeader*.
- For each original *Package* that uses the standard *PackageHeader,* the corresponding *Package* in the delivered *TmNSDataMessage* shall have a *Package Time* equal to the *Package Time* of the original *Package* according to the *PackageDefinition* corresponding to the **TmNSdeliverypdid**.

26.4.1.5.3.2 MeasurementID Request Non-Standard PackageHeader Handling

RTSPDataSources processing a *MeasurementID* request and that support extraction from *Packages* that do not use the standard *PackageHeader* shall adhere to the following requirement:

- The *RTSPDataSource* may deliver some or all requested *MeasurementData* from original *Packages* that do not use the standard *PackageHeader.*
- For each original *Package* that does not use the standard *PackageHeader*, the corresponding *Package* in the delivered *TmNSDataMessage* shall have a *Package Time* equal to the *Message Time* of the original *Package* according to the *PackageDefinition* corresponding to the **TmNSdeliverypdid**.

26.4.2 RTSP-Based Data Channel (*RTSPDataChannel*)

The operation of the *RTSPDataChannel* shall be controlled by the *RTSPControlChannel* as specified in Section 26.4.1. The *RTSPDataChannel* transport protocol (TCP or UDP) is specified in the Transport Header of the *DataRequest*.

RTSPDataChannel messages shall use the standard *TmNSDataMessage* structure and mechanisms as specified in Chapter 24, Message Formats.

Upon receipt of a valid SETUP request, an *RTSPDataSource* shall open the *RTSPDataChannel* socket.

Upon receipt of a valid PLAY request, an *RTSPDataSource* shall attempt to transmit requested data to the *RTSPDataChannel* socket.

Upon receipt of a TEARDOWN request, an *RTSPDataSource* shall close the *RTSPDataChannel* socket.

After receiving the TEARDOWN response, an *RTSPDataSink* shall close the *RTSPDataChannel* socket.

NOTE	Pausing and then resuming a UDP *RTSPDataChannel* may result in data loss. Pausing and then resuming a TCP *RTSPDataChannel* should not result in data.

NOTE	Handling data loss on a *DataChannel* is not addressed by this standard.

26.4.2.1 TCP-Based *RTSPDataChannel*

Prior to issuing a SETUP request, an *RTSPDataSink* shall open the *RTSPDataChannel* socket. The *RTSPDataSink* shall execute a listen on the socket and optionally obtain an ephemeral TCP port number (which would be included in the Transport Header).

Upon receipt of a SETUP request, an *RTSPDataSource* shall execute a connect on the socket (the SETUP request's Transport Header contains the transport protocol information).

26.4.2.2 End of Data Indication

When the *RTSPDataSource* is ready to close the *RTSPDataChannel*, the *RTSPDataSource* shall deliver an End of Data Indicator to the *RTSPDataSink*.

The *RTSPDataSource* may set the **EndOfDataFlag** in the *TmNSDataMessageHeader* of the last *TmNSDataMessage* prior to sending the last *TmNSDataMessage* to the *RTSPDataSink*. Alternatively, or if no *TmNSDataMessages* have been sent, the *RTSPDataSource* shall deliver a *TmNSDataMessage* with no *TmNSDataMessagePayload* and the following values in the *TmNSDataMessageHeader*:

- Set MessageFlags to 16'h0001, which sets only the EndOfDataFlag
- Set MessageDefinitionID to 32'd0.
- Set MessageDefinitionSequenceNumber to 32'd0.
- Set MessageLength to 32'd24.
- Set MessageTimestamp to 64'd0.

26.4.3 Reliability Critical (RC) Delivery Protocol

The *RC Delivery Protocol* is the *TmNS*-specific application-level method of delivering *TmNSDataMessages* via TCP.

NOTE	The *RC Delivery Protocol* section and all related subsections specify how to deliver *TmNSDataMessages* when reliability of data delivery is more important than low latency or high throughput.

26.4.3.1 RC Delivery Protocol Data Channel (*RCDataChannel*)

RCDataSources and *RCDataSinks* shall support the *RTSPDataChannel* as defined in Section 26.4.2.

RCDataSources shall transport *TmNSDataMessages* to the *RCDataSink*'s IP address and the destination port specified in the Transport Header.

26.4.3.2 RC Delivery Protocol Control Channel (*RCControlChannel*)

RCDataSources and *RCDataSinks* shall exchange control commands and parameters using the *RTSPControlChannel*, as defined in Section 26.4.1. This section specifies additional constraints on using the *RTSPControlChannel* as the *RCControlChannel*.

The RTSP "Transport" header shall specify TCP, which shall be used for the transport of *TmNSDataMessages* on the *RCDataChannel*.

A *DataRequest* from an *RCDataSink* shall use at least one of the following three request types as specified in Sections 26.4.1.5:

- MessageDefinitionID request
- PackageDefinitionID request
- MeasurementID request

26.4.4 Request-Defined Data Channel

NOTE	An HTTP-based Data Request interface has been proposed to the RCC-TG and is currently under consideration as a possible enhancement to Request-Defined Data Transfers.

26.5 *TmNSDataMessage* Transfer Rules

DataSources and *DataSinks* shall comply with the standard *TmNSDataMessage* structure and mechanisms, as specified in Chapter 24, Message Formats. *DataSources* shall adhere to the following *TmNSDataMessage* transfer rules:

1. Multiple sequences of *TmNSDataMessages* that contain different *MessageDefinitionIDs* may be sent to the same multicast or unicast destination address.

2. Multiple *DataSources* shall not send *TmNSDataMessages* with the same *MessageDefinitionID* to the same destination address unless the multiple *DataSources* synchronize the incrementing of the **MessageSequenceNumber** field in accordance with the Sequence Number Convention specified in Section 26.5.1.

3. Replicated *TmNSDataMessages* may be sent to multiple destination addresses provided rule 2 above is not violated.

NOTE	When adding *Packages* to the acquisition *TmNSDataMessage* Payload, a *DataSource* should use a mechanism taking the minimum of "maximum message size" and "maximum elapsed time" variables to determine when to send a complete *TmNSDataMessage* of sampled data.

26.5.1 Sequence Numbering Convention

Each *TmNSDataMessageHeader* contains a **MessageSequenceNumber** field whose value increments by one for each *TmNSDataMessage* instance in a sequence of *TmNSDataMessages*. The **MessageSequenceNumber** value shall wrap to zero after $2^{32} – 1$. The wrapping of the **MessageSequenceNumber** value to zero shall not indicate a loss.

For *DataSources* generating *TmNSDataMessages*, **MessageSequenceNumber** values are assigned on a per-*MessageDefinitionID* basis.

The **MessageSequenceNumber** value shall not repeat consecutively or be generated out of order for a particular sequence of *TmNSDataMessages*, including when two or more *DataSources* generate *TmNSDataMessages* with the same *MessageDefinitionID*.

The **MessageSequenceNumber** field for a *TmNSDataMessage* sequence shall be set to zero upon:

- The power-up or reset of the *NetworkNode* generating the corresponding *TmNSDataMessage* sequence.
- The configuration, reconfiguration, or reset of the *TmNSApp* generating the corresponding *TmNSDataMessage* sequence.
- The instantiation of a Request-Defined *DataChannel* generating the corresponding *TmNSDataMessage* sequence.

26.5.2 Timestamp Convention

The **MessageTimestamp** value of a given *TmNSDataMessage* shall be no earlier than all of the acquisition times of *MeasurementData* samples in the previous *TmNSDataMessage* instance in the sequence of *TmNSDataMessages*. See Section 26.5.1, Sequence Numbering Convention, for the description of a sequence of *TmNSDataMessages*.

26.5.3 TmNSDataMessage Fragmentation

TmNSDataMessages support being broken up into multiple fragments. The **MessageFragmentationFlags** of the *TmNSDataMessageHeader* identify how to reconstruct a full *TmNSDataMessage*. All fragments of a *TmNSDataMessage* shall include the same value for the **MessageTimestamp** and **MessageFlags** fields in their *TmNSDataMessageHeader* with the following exception for the **MessageFragmentationFlags** bits:

- The first fragment shall set the **MessageFragmentationFlags** bits to "2'b01" (*TmNSDataMessage* with the first fragment).
- Each middle fragment shall set the **MessageFragmentationFlags** bits to "2'b10" (*TmNSDataMessage* with a middle fragment).
- The last fragment shall set the **MessageFragmentationFlags** bits to "2'b11" (*TmNSDataMessage* with the last fragment).

Each fragment's **MessageSequenceNumber** field value shall follow the sequence numbering convention as described in Section 26.5.1.

26.5.4 Generating *TmNSDataMessages* from Other *TmNSDataMessages* Convention

DataSources that combine data from multiple *TmNSDataMessages* into a new *TmNSDataMessage* shall bitwise-OR the following *DataSource*-specific **MessageFlags** from the original *TmNSDataMessages* to form the resultant *DataSource*-specific **MessageFlags**:

DataSourceHealthFlag
DataSourceTimeLockFlag
DataSourceAcquiredDataFlag

Any **ApplicationDefinedFields** in the transferred *TmNSDataMessages* shall indicate conditions on the *DataSource* delivering the *TmNSDataMessages*, not the original *DataSource* (the **ApplicationDefinedFields** from the original *TmNSDataMessages* are discarded).