INTERNATIONAL STANDARD

ISO/IEC 23009-5

First edition
2017-02

Information technology — Dynamic adaptive streaming over HTTP (DASH) —

Part 5: Server and network assisted DASH (SAND)

Technologies de l'information — Diffusion en flux adaptatif dynamique sur HTTP (DASH) —

Partie 5: DASH assisté par serveur et réseau (SAND)

Reference number
ISO/IEC 23009-5:2017(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword ..... **v**
Introduction ..... **vi**
1 Scope **1**
2 Normative references **1**
3 Terms, definitions, symbols and abbreviated terms **1**
3.1 Terms and definitions 1
3.2 Abbreviated terms 2
3.3 Conventions 2
4 Overview **3**
5 SAND reference architecture and interfaces **3**
6 SAND messages **7**
6.1 General 7
6.2 Common Envelope for SAND messages 7
6.3 Metrics messages 9
6.3.1 General 9
6.3.2 TCPConnections 9
6.3.3 HTTPRequestResponseTransactions 9
6.3.4 RepresentationSwitchEvents 9
6.3.5 BufferLevel 10
6.3.6 PlayList 10
6.4 Status Messages 11
6.4.1 AnticipatedRequests 11
6.4.2 SharedResourceAllocation 11
6.4.3 AcceptedAlternatives 12
6.4.4 AbsoluteDeadline 13
6.4.5 MaxRTT 14
6.4.6 NextAlternatives 14
6.4.7 ClientCapabilities 15
6.5 PER Messages 16
6.5.1 ResourceStatus 16
6.5.2 DaneResourceStatus 17
6.5.3 SharedResourceAssignment 19
6.5.4 MPDValidityEndTime 19
6.5.5 Throughput 20
6.5.6 AvailabilityTimeOffset 21
6.5.7 QoSInformation 22
6.5.8 DeliveredAlternative 23
6.5.9 DaneCapabilities 24
6.6 PED Messages 25
7 SAND message representation format **25**
8 Transport Protocol to carry SAND messages **25**
8.1 General 25
8.2 Protocol to carry metrics and status messages 25
8.2.1 General 25
8.2.2 Sending a message directly to Metrics server or DANE 26
8.2.3 Attaching a message to requests for media 26
8.3 Protocol to carry PER messages 27
8.3.1 General 27
8.3.2 Assistance 28
8.3.3 Enforcement 28
8.3.4 Error case 28

9 Signalling of SAND communication channel 28
9.1 General 28
9.2 XML schema for `sand:Channel` element 29

10 Optional transport protocols to carry SAND messages 30
10.1 General 30
10.2 WebSocket protocol 30
10.2.1 General 30
10.2.2 Signalling via the MPD 31
10.2.3 WebSocket messages 32

11 Reporting of metrics via SAND protocols 32

Annex A (normative) **XML Schema for SAND messages 33**

Annex B (normative) **SharedResourceAllocation allocation strategies 41**

Annex C (normative) **MIME type registration for SAND message 45**

Bibliography 47

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

A list of all parts in the ISO/IEC 23009 series can be found on the ISO website.

Introduction

In order to enhance the delivery of DASH content, this document introduces messages between DASH clients and network elements or between various network elements for the purpose of improving the efficiency of streaming sessions by providing information about real-time operational characteristics of networks, servers, proxies, caches, CDNs, as well as DASH client's performance and status.

Information technology — Dynamic adaptive streaming over HTTP (DASH) —

Part 5:
Server and network assisted DASH (SAND)

1 Scope

This document defines the following:

— the functional SAND architecture which identifies the SAND network elements and the nature of SAND messages exchanged among them;

— the semantics of SAND messages exchanged between the network elements present in the SAND architecture;

— an encoding scheme for the SAND messages;

— the SAND message delivery protocol.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 8601, *Data elements and interchange formats — Information interchange — Representation of dates and times*

ISO/IEC 23009-1:2014, *Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats*

IETF RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*

IETF RFC 6455, *The WebSocket Protocol*

IETF RFC 7233:2014, *Hypertext Transfer Protocol (HTTP/1.1): Range Requests*

3 Terms, definitions, abbreviated terms and conventions

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 23009-1 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at http://www.iso.org/obp

3.1.1
DASH Aware Network Element
DANE
network element which has at least minimum intelligence about DASH; for instance, it may be aware that the delivered objects are DASH-formatted objects such as the MPD or DASH segments, and may prioritize, parse or even modify such objects

3.1.2
SAND messages
messages exchanged between DASH clients, DASH aware Network Elements or Metrics Server in order to either enhance reception (PER) or delivery (PED) of DASH service, or to report status or metrics from the DASH client to DASH aware Network Elements or Metrics Server

3.2 Abbreviated terms

DANE	DASH aware network element
DASH	Dynamic Adaptive Streaming over HTTP
DM	DASH Metrics
HTTP	Hypertext Transfer Protocol
MPD	Media Presentation Description
PED	parameters enhancing delivery
PER	parameters enhancing reception
RNE	regular network element
SAND	server and network assisted DASH
TLS	transport layer security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UTC	Coordinated Universal Time
UTF	unicode transformation format
UUID	universally unique identifier
XML	Extensible Mark-Up Language

3.3 Conventions

The following naming conventions apply in this document.

— Elements in an XML document are identified by an upper-case first letter and in bold face as **`Element`**. To express that an element **`Element1`** is contained in another element **`Element2`**, we may write **`Element2.Element1`**. If an element's name consists of two or more combined words, camel-casing is typically used, e.g. **`ImportantElement`**. Elements may be present either exactly once, or the minimum and maximum occurrence is defined by <minOccurs> ... <maxOccurs>.

— Attributes in an XML document are identified by a lower-case first letter, as well as they are preceded by a '`@`'-sign, e.g. `@attribute`. To point to a specific attribute `@attribute` contained

in an element **Element**, one may write **Element**`@attribute`. If an attribute's name consists of two or more combined words, camel-casing is typically used after the first word, e.g. `@veryImportantAttribute`. Attributes may have assigned a status in the XML as mandatory (M), optional (O), optional with default value (OD) and conditionally mandatory (CM).

— Namespace qualification of elements and attributes is used as per XML standards, in the form of **`namespace:Element`** or `@namespace:attribute`. The fully qualified namespace will be provided in the schema fragment associated with the declaration. External specifications extending the namespace of DASH are expected to document the element name in the semantic table with an extension namespace prefix.

— Variables defined in the context of this document are specifically highlighted with *italics,* e.g. *InternalVariable*.

— Structures that are defined as part of the hierarchical data model are identified by an upper-case first letter, e.g. Period, Adaptation Set, Representation, Segment, etc.

— The term "this clause" refers to the entire clause included within the same first heading number. The term "this subclause" refers to all text contained in the subclause with the lowest hierarchy heading.

4 Overview

In recent years, the Internet has become an important channel for the delivery of multimedia using HTTP as its primary protocol. In 2014, ISO/IEC published the second edition of MPEG Dynamic Adaptive Streaming over HTTP (DASH) as an International Standard that specified formats for the media presentation description (MPD), as well as ISO-BMFF and MPEG-2 TS based segments. DASH does not define a system or protocol, but is considered as an enabler for efficient and high-quality delivery of multimedia content over the Internet.

In order to enhance the delivery of DASH content, this document introduces messages between DASH clients and network elements or between various network elements for the purpose of improving efficiency of streaming sessions by providing information about real-time operational characteristics of networks, servers, proxies, caches, CDNs as well as DASH client's performance and status.

The Server and network assisted DASH (SAND) addresses the following:

— unidirectional/bidirectional, point-to-point/multipoint communication with and without session (management) between servers/CDNs and DASH clients;

— mechanisms for providing content-awareness and service-awareness towards the underlying protocol stack including server and/or network assistance;

— various impacts on elements of the existing Internet infrastructure such as servers, proxies, caches and CDNs;

— QoS and QoE support for DASH-based services;

— scalability in general and specifically for logging interfaces;

— analytics and monitoring of DASH-based services.

5 SAND reference architecture and interfaces

The SAND reference architecture is based on the following four broad categories of elements.

a) DASH clients.

b) Regular network elements (RNE), which are DASH unaware and treat DASH delivery objects as any other object, but are present on the path between origin server and DASH clients, e.g. transparent caches. Note that such regular network elements are not in the scope of this document.

c) DASH aware network elements (DANE), which have at least minimum intelligence about DASH; for instance, they may be aware that the delivered objects are DASH-formatted objects such as the MPD or DASH segments, and may prioritize, parse or even modify such objects. More details on typical DANE functionalities are provided.

d) Metrics server, which are DASH aware and are in charge of gathering metrics from DASH clients.

Based on these elements, the SAND reference architecture is defined as shown in Figure 2. Within this architecture, the following four categories of messages, called SAND messages as shown in Figure 1, are exchanged:

— Parameters Enhancing Delivery (PED) messages that are exchanged between DANEs;

— Parameters Enhancing Reception (PER) messages that are sent from DANEs to DASH clients;

— status messages that are sent from DASH clients to DANEs;

— metrics messages that are sent from DASH clients to Metrics servers.

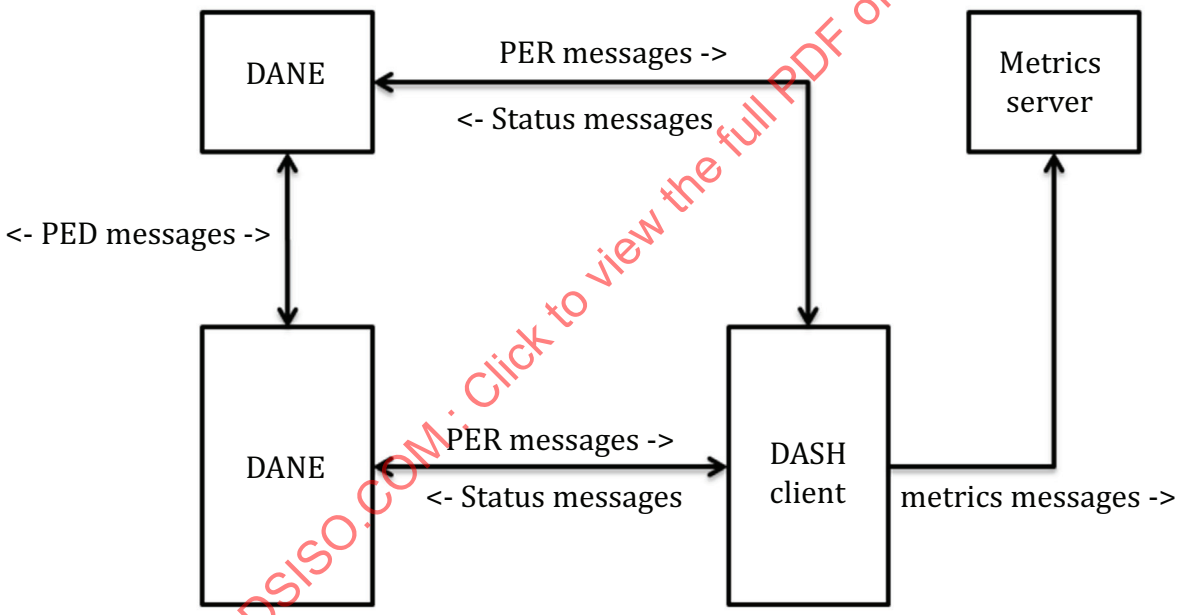


Figure 1 — SAND messages

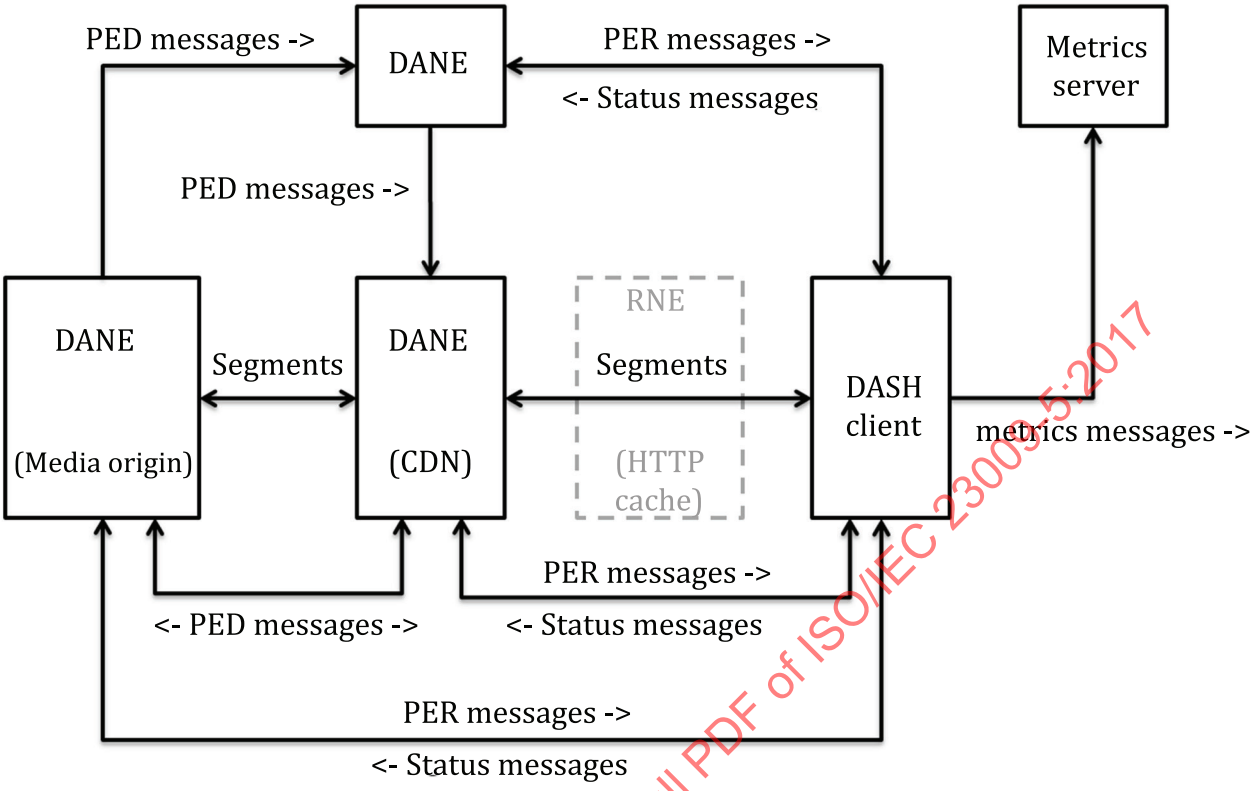


Figure 2 — SAND reference architecture

In this context, a media origin that serves DASH content may also receive status messages from the clients, send PED parameters to other DANEs, and is therefore also considered as a DANE element.

Similarly, a third-party server that may receive SAND status messages from DASH clients or send SAND PER messages to the clients is considered as a DANE element. Note that the third-party server may not necessarily be on the media delivery path and it may not see the DASH segments. However, as it may understand the SAND status messages or produce SAND PER messages to DASH clients, e.g to improve delivery efficiency, it is nevertheless considered as a DANE element.

A DASH client may send two types of messages: metric messages carrying metric information and status messages carrying operational information. The metrics and status messages have a similar structure; however, it is important to distinguish them since these messages carry information of different nature. Whereas status message provide real-time feedback from the client to DANEs in order to support real-time operations, Metrics are provided as a summary of the session or at least longer time intervals of the session and are not considered provided in real-time.

Based on this terminology, the following interfaces are considered:

— Client-to-Metrics-server Interface: Carries metrics messages;

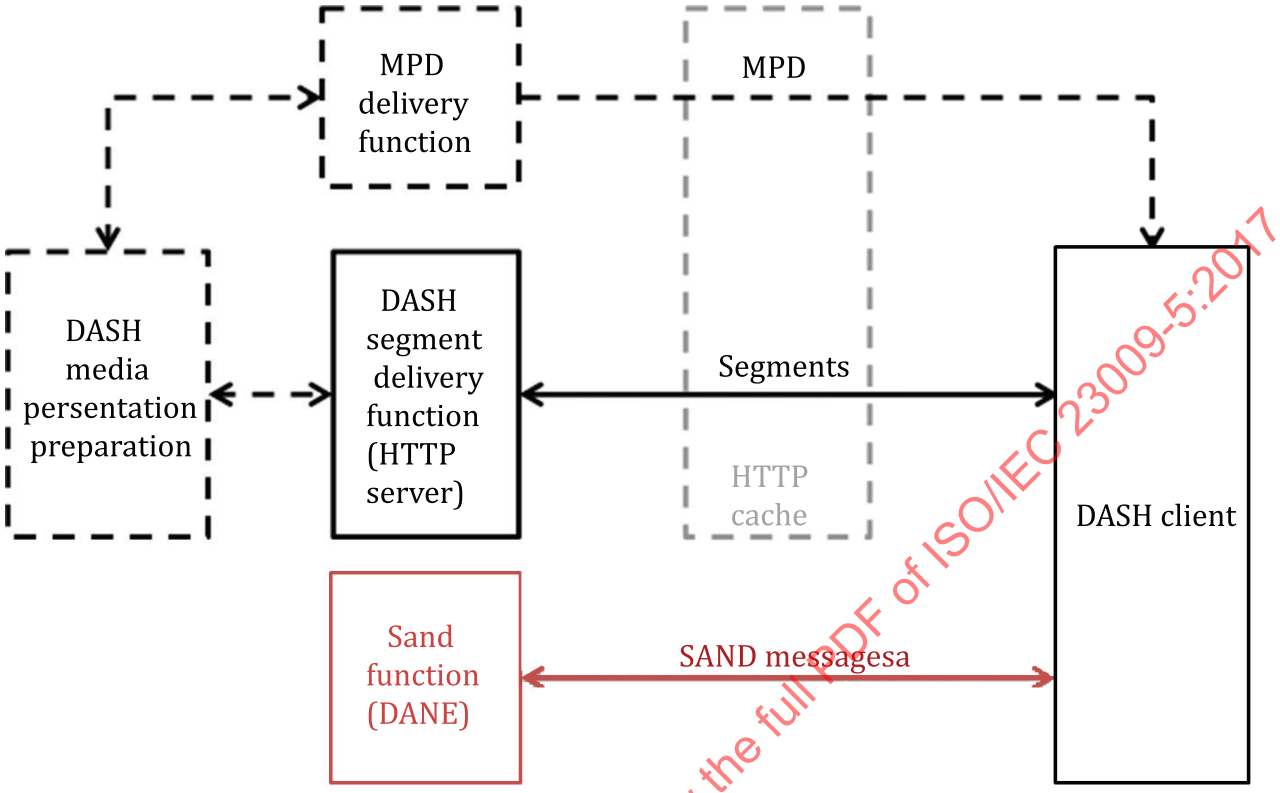
— Client-to-DANE Interface: Carries status messages;

— DANE-to-DANE Interface: Carries PED messages;

— DANE-to-Client Interface: Carries PER messages.

The implementation of the SAND architecture is neither mandatory nor necessary for successful DASH-based streaming operation. One may choose to implement a subset of the interfaces and messages defined in the SAND reference architecture.

ISO/IEC 23009-1 gives also an overview of a possible deployment architecture for DASH. In light of the SAND reference architecture above, Figure 3 suggests a possible extension to it for a SAND-augmented DASH architecture.



Key

a PER, metrics and status messages.

Figure 3 — SAND-augmented DASH architecture

In ISO/IEC 23009-1, the DASH client model consists of the DASH access engine, the Media engine and the Application. The DASH access engine operates at the interface with the network when it comes to receiving the MPD and the segments, even though the delivery of the MPD is out-of-scope of MPEG DASH as stated in ISO/IEC 23009-1:2014, 5.2.1. To support the interfaces in the SAND reference architecture, the DASH access engine becomes also responsible for the communication with the DANEs since the DANEs are network elements providing DASH-related information. Figure 4 extends the original DASH Client model from ISO/IEC 23009-1 with the addition of a SAND channel to communicate with DANEs.

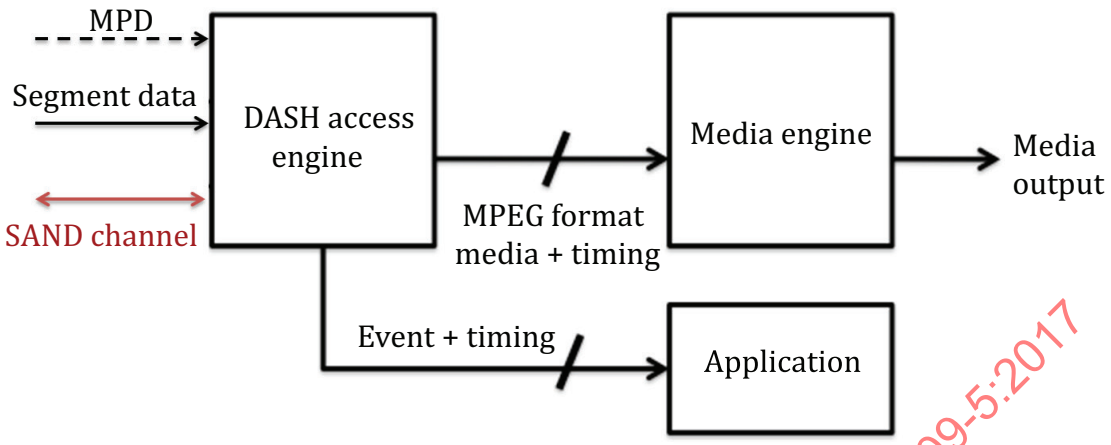


Figure 4 — SAND-augmented DASH Client model

6 SAND messages

6.1 General

This clause defines the SAND messages and their attributes. The format representation of SAND messages is XML and is defined in Clause 7. SAND messages are XML documents for which the MIME type is defined in Annex C.

SAND implementations may choose to support all or only a subset of the SAND messages defined here. Additionally, implementations may create their own messages by using their own XML namespace.

SAND elements (DANEs and DASH clients) are expected to not send SAND messages when there is no entity in the network for using them.

6.2 Common Envelope for SAND messages

All SAND messages shall use the common envelope of Table 1.

For better efficiency, the common envelope for SAND messages allows SAND messages aggregation within the same envelope. The envelope is the unit of data that is provided to the transport layer for sending SAND message.

All SAND messages have many parameters in common. Parameters which convey the same value for all messages included in a same envelope are attached to this common envelope for SAND messages. All messages also share common parameters that need to be assigned different values for each individual message in an envelope. Table 1 defines these parameters and is implicitly included in all of the following message definitions.

Within this document, the date-time type indicated in the tables shall be represented as specified by ISO 8601.

Table 1 — SAND message common envelope

Parameter	Type	Cardinality	Description
CommonEnvelope		1	
senderId	token	0..1	If present, this is a unique identifier of the message sender. It is up to the sender to provide such a unique identifier.
generationTime	date-time	0..1	If present, this indicates the UTC time (format as specified by ISO 8601) at which the message was generated.

Table 1 *(continued)*

Parameter	Type	Cardinality	Description
SandMessage		1..N	
messageId	int	0..1	This field allows receivers of SAND messages to discriminate between several messages sent from the same sender. Identification of the sender may be done thanks to the senderId information or other transport layer information if senderId is not present. Among messages with same sender and same messageType, message with highest messageId value shall take precedence over the others in case of any conflict between the information they convey. The maximum value for messageId is decided by senders and shall be high enough for receivers to easily identify which message shall take precedence even when messageId values have looped back to 0.
validityTime	date-time	0..1	If present, this indicates the UTC time after which the validity of the message is not guaranteed anymore. If not present, validity of messages lasts until next message carrying conflicting information with same sender, same messageType and higher messageId is received.

Table 2 — messageType values

messageType	Message description
0	reserved
1	TCPConnections (6.3.2)
2	HTTPRequestResponseTransactions (6.3.3)
3	RepresentationSwitchEvents (6.3.4)
4	BufferLevel (6.3.5)
5	PlayList (6.3.6)
6	AnticipatedRequests (6.4.1)
7	SharedResourceAllocation (6.4.2)
8	AcceptedAlternatives (6.4.3)
9	AbsoluteDeadline (6.4.4)
10	MaxRTT (6.4.5)
11	NextAlternatives (6.4.6)
12	ClientCapabilities (6.4.7)
13	ResourceStatus (6.5.1)
14	DaneResourceStatus (6.5.2)
15	SharedResourceAssignment (0)
16	MPDValidityEndTime (6.5.4)
17	Throughput (6.5.5)
18	AvailabilityTimeOffset (6.5.6)
19	QoSInformation (6.5.7)
20	DeliveredAlternative (6.5.8)
21	DaneCapabilities (6.5.9)
22..127	reserved for future ISO use
128..255	reserved for private use

The type of SAND messages is uniquely identified thanks to a messageType value. Allowed values for this field are described in Table 2. When using XML representation format, this field shall be represented by

the tag name of the message element. When using the HTTP header transport of 8.2.3, this field shall be represented as the header name.

6.3 Metrics messages

6.3.1 General

This subclause defines the XML format of the DASH metrics defined in ISO/IEC 23009-1:2014, Annex D. This enables the reporting of the DASH metrics over the SAND framework.

6.3.2 TCPConnections

6.3.2.1 Motivation

This metric collects information at the TCP level about HTTP request/response transactions.

6.3.2.2 Source and destination

```
Type     : Metrics
Sender   : DASH client
Receiver : DANE
```

6.3.2.3 Data representation

See TCP connections in ISO/IEC 23009-1:2014, D.4.2 for the semantics.

6.3.3 HTTPRequestResponseTransactions

6.3.3.1 Motivation

This metric collects information about HTTP request/response transactions.

6.3.3.2 Source and destination

```
Type : Metrics
Sender : DASH client
Receiver : DANE
```

6.3.3.3 Data representation

See HTTP request/response transactions in ISO/IEC 23009-1:2014, D.4.3 for the semantics.

6.3.4 RepresentationSwitchEvents

6.3.4.1 Motivation

This metric collects information describing Representation switch events.

6.3.4.2 Source and destination

```
Type : Metrics
Sender : DASH client
Receiver : DANE
```

6.3.4.3 Data representation

See Representation switch events in ISO/IEC 23009-1:2014, D.4.4 for the semantics.

6.3.5 BufferLevel

6.3.5.1 Motivation

This metric reflects the Buffer level at a given point in time.

6.3.5.2 Source and destination

```
Type : Metrics
Sender : DASH client
Receiver : DANE
```

6.3.5.3 Data representation

See Buffer level in ISO/IEC 23009-1:2014, D.4.5 for the semantics.

6.3.6 PlayList

6.3.6.1 Motivation

This metric collects information about playback periods.

6.3.6.2 Source and destination

```
Type : Metrics
Sender : DASH client
Receiver : DANE
```

6.3.6.3 Data representation

See Play list in ISO/IEC 23009-1:2014, D.4.6 for the semantics.

6.4 Status Messages

6.4.1 AnticipatedRequests

6.4.1.1 Motivation

This message allows a DASH client to announce to a DANE which specific set of segments it is interested in. The intent is to signal the set of segments in representations that the DASH client is likely to select and request soon.

NOTE The validity duration of this message is given by the validityTime parameter in the message envelope (see Table 1).

6.4.1.2 Source and destination

```
Type : Status
Sender : DASH client
Receiver : DANE
```

6.4.1.3 Data representation

Table 3 shows the AnticipatedRequest parameters.

Table 3 — AnticipatedRequests parameters

Parameter	Type	Cardinality	Description
AnticipatedRequests	object	1	
request	array	1..N	List of anticipated requests
sourceUrl	url	1	URL for a segment of a given representation.
range	string	0..1	This is the byte range specification when the segment is only a part of the content referred to by sourceUrl.
targetTime	date-time	0..1	Time at which the DASH client expects to request the resource identified by sourceUrl.

6.4.2 SharedResourceAllocation

6.4.2.1 Motivation

This message groups all information allowing a DASH client to indicate to one or several DANE(s) an intent to share network resources (for example, access link bandwidth in a home network).

NOTE The validity duration of this message is given by the validityTime parameter in the message envelope (see Table 1).

6.4.2.2 Source and destination

```
Type: : Status
Sender : DASH client
Receiver : DANE
```

6.4.2.3 Data representation

Table 4 shows the SharedResourceAllocation parameters.

Table 4 — SharedResourceAllocation parameters

Parameter	Type	Cardinality	Description
SharedResourceAllocation	object	1	
operationPoints	array	1..N	List of suitable operation points for current play time.
bandwidth	integer	1	A bandwidth value expressed in a number of bits/s. This value should be computed from the MPD by summing bandwidths of all components the client would use for working at this operation point. If playback rate is not 1, this bandwidth value shall be modified accordingly.
quality	integer	0..1	An optional value describing the quality of the current operation point. NOTE Quality's scale is left to implementation.
minBufferTime	integer	0..1	An optional value in milliseconds extracted from the MPD regarding a minimal buffer time of the current operation point.
weight	integer	0..1	A user allocated optional value which indicates a weight of the request in the present message for the resource allocation process. The exact use of this value depends on the allocation strategy indicated by allocationStrategy.
allocationStrategy	urn	0..1	An optional identifier to indicate the resource allocation strategy preferred by the client for resource sharing. See Annex B. If absent, the value `urn:mpeg:dash:sand:allocation:basic:2016` is assumed.
mpdUrl	url	0..1	If present, the MPD's URL corresponding to the present message.

6.4.3 AcceptedAlternatives

6.4.3.1 Motivation

This message allows DASH clients to inform DANEs on the media delivery path (typically caching DANEs) when they request a given DASH segment that they are willing to accept other DASH segments alternatives. A client should not include alternative segments unless it is ready to receive them and be able to play them.

NOTE The validity duration of this message is given by the validityTime parameter in the message envelope (see Table 1).

6.4.3.2 Source and destination

```
Type: : Status
Sender : DASH client
Receiver : DANE
```

6.4.3.3 Data representation

Table 5 shows the AcceptedAlternatives parameters.

Table 5 — AcceptedAlternatives parameters

Parameter	Type	Cardinality	Description
AcceptedAlternatives	array	1..N	The ordered list of acceptable alternatives. Preferred alternatives are listed first.
alternative	object	1	Specification of one acceptable alternative.
sourceUrl	url	1	This is the URL of the alternative, as deduced from the MPD@sourceUrl for requesting the (sub)segment of the acceptable representation. If no protocol is present at the beginning of the URL, then it is a relative URL with regards to the URL of the segment request to which this message is attached.
range	string	0..1	This is the byte range specification when the segment is only a part of the content referred to by sourceUrl. It has the same syntax as the @range attribute of an URLType as specified in ISO/IEC 23009-1.
bandwidth	integer	0..1	Optional bandwidth in bits/s that is considered as necessary by the client to receive the alternative in good conditions.
deliveryScope	integer	0..1	Optional parameter that indicates the number of caching DANEs that may be reached before removing this alternative from the list when forwarding the request. NOTE DANEs are expected to decrement this counter when forwarding the request and remove the alternative from the list when counter reaches 0.

6.4.4 AbsoluteDeadline

6.4.4.1 Motivation

This message allows DASH clients indicating the DANE the absolute deadline in wall-clock time by when the requested segment needs to be completely received.

AbsoluteDeadline message shall not be sent using the HTTP POST method.

NOTE The validity duration of this message is given by the validityTime parameter in the message envelope (see Table 1).

6.4.4.2 Source and destination

```
Type: : Status
Sender : DASH client
Receiver : DANE
```

6.4.4.3 Data representation

Table 6 shows the AbsoluteDeadline parameters.

Table 6 — AbsoluteDeadline parameters

Parameter		Type	Cardinality	Description
AbsoluteDeadline		object	1	
	deadline	date-time	1	Absolute deadline for the segment to be available in the receiver.

6.4.5 MaxRTT

6.4.5.1 Motivation

This message allows DASH clients indicating the DANE the maximum round trip time of the request from the time when the request was issued until the request needs to be completely available at the DASH client. The time is expressed in milliseconds.

If MaxRTT message is sent with HTTP POST method, it shall be intended for every segment request during validity time.

NOTE The validity duration of this message is given by the validityTime parameter in the message envelope (see Table 1).

6.4.5.2 Source and destination

```
Type: : Status
Sender : DASH client
Receiver : DANE
```

6.4.5.3 Data representation

Table 7 shows the MaxRTT parameters.

Table 7 — MaxRTT parameters

Parameter		Type	Cardinality	Description
MaxRTT		object	1	
	maxRTT	int	1	Maximum RTT from the request until the Segments is available in milliseconds.

6.4.6 NextAlternatives

6.4.6.1 Motivation

This message allows DASH clients to inform a DANE about which alternatives they are willing to accept for the request of the next segment.

In response to this message, a DANE on the media delivery path (typically caching DANE) may provide an assistance PER message through the notification mechanism of 8.3.2. A DANE supporting this feature should notify in the response headers a URL which corresponds to a DaneResourceStatus message with relevant information.

NOTE The validity duration of this message is given by the validityTime parameter in the message envelope (see Table 1).

6.4.6.2 Source and destination

```
Type: : Status
Sender : DASH client
Receiver : DANE
```

6.4.6.3 Data representation

Table 8 shows the NextAlternatives parameters.

Table 8 — NextAlternatives parameters

Parameter	Type	Cardinality	Description
NextAlternatives	array	1..N	The ordered list of alternatives the client is willing to accept for the next segment. Preferred alternatives are listed first.
alternative	object	1	Specification of one acceptable alternative.
sourceUrl	url	1	This is the URL of the alternative, as deduced from the MPD @sourceUrl for requesting the (sub)segment of the acceptable representation.
range	string	0..1	This is the byte range specification when the segment is only a part of the content referred to by sourceUrl. It has the same syntax as the @range attribute of an URLType as specified in ISO/IEC 23009-1.
bandwidth	integer	0..1	Optional bandwidth in bits/s that is considered as necessary by the client to receive the alternative in good conditions.
deliveryScope	integer	0..1	Optional parameter that indicates the number of caching DANEs that may be reached before removing this alternative from the list when forwarding the request. NOTE DANEs are expected to decrement this counter when forwarding the request and remove the alternative from the list when counter reaches 0.

6.4.7 ClientCapabilities

6.4.7.1 Motivation

This message allows a DASH client to announce to a DANE which specific SAND messages it supports (both as a receiver and emitter) by listing the MessageType (see Table 2) of all supported SAND messages. In addition, a message set URN that regroups several SAND messages together may also be used to signal additional support of a group of SAND messages.

NOTE The validity duration of this message is given by the validityTime parameter in the message envelope (see Table 1).

6.4.7.2 Source and destination

```
Type : Status
Sender : DASH client
Receiver : DANE
```

6.4.7.3 Data representation

Table 9 shows the ClientCapabilities parameters.

Table 9 — ClientCapabilities parameters

Parameter	Type	Cardinality	Description
ClientCapabilities	object	1	Provides the list of SAND messages supported by the DASH client. A profile may also be used.
supportedMessage	array	0..N	Provides the list of SAND messages supported by the DASH client.
messageType	int	1	MessageType of one supported SAND message (see Table 2).
messageSetUri	urn	0..1	URN of a message set that regroups several SAND messages together.

The only message set URN defined in the present specification is `urn:mpeg:dash:sand:messageset:all:2016`. DASH clients which supports this message set actually support all SAND messages defines in present specification.

6.5 PER Messages

6.5.1 ResourceStatus

6.5.1.1 Motivation

This message allows for a DANE to inform a DASH client, typically in advance, about knowledge of segment availability including the caching status of the segment(s) in the DANE. The status may be different for different baseUrl or different Representation ID (repId) used, allowing to signal availability of content dependent on the network delivering it. This may be, for example, because the segments are being delivered over a certain network and are expected to be selected by the DASH client regardless of the decision of the rate adaptation algorithm.

The message expresses the status of the segment availability at the current time. The DASH client should assume that this status persists until it is informed about a change of the status or the validityTime of the message is passed (see Table 1).

6.5.1.2 Source and destination

```
Type : PER
Sender : DANE
Receiver : DASH client
```

6.5.1.3 Data representation

Table 10 shows the ResourceStatus (with baseUrl) parameters while Table 11 shows ResourceStatus (with representataion ID) parameters.

Table 10 — ResourceStatus (with baseUrl) parameters

Parameter	Type	Cardinality	Description
ResourceStatus	object	1	Resource Status Information for resources identified from a base URL.
resourceInfo	array	1..N	List of Resource Status Information.
baseUrl	url	1	Provides the base URL for the associated resources, i.e. the status holds for all resources referenced by this base URL.
status	enum	1	Provides the status of all associated resource to the base URL. The defined types are documented in Table 12.
reason	string	0..1	Provides some textual information of the reason, e.g. 'you are in broadcast mode'.

Table 11 — ResourceStatus (with representation ID) parameters

Parameter	Type	Cardinality	Description
ResourceStatus	object	1	Resource Status Information for segments identified from a representation ID.
resourceInfo	array	1..N	List of Resource Status Information.
repId	string	1	Provides the value for the representation ID, i.e. status holds for all resources associated to a Representation with the value of the parameter.
status	enum	1	Provides the status of all associated resource to the Representation. The defined types are documented in Table 12.
reason	string	0..1	Provides some textual information of the reason, e.g. "you are in broadcast mode".

Table 12 — Allowed values for status parameter

Status	Semantics
available	Resource is available in DANE and request is expected to be responded with a 2xx code.
unavailable	Resource is not available in the DANE and request is expected to be responded with a 4xx code.
cached	Resource will be available in the DANE at the time announced in the Media Presentation.

6.5.2 DaneResourceStatus

6.5.2.1 Motivation

This parameter allows DANEs to signal the available and possibly anticipated to be available data structures to the DASH client and also signal which data structures are unavailable. This method is complementary to the resource status mentioned above as it allows to express the available segments at the time of the status message. The resources are either explicitly listed or provided as a list, or they are provided by some abbreviated message format.

The message expresses the status of the segment availability at the current time. The DASH client should assume that this status persists until it is informed about a change of the status or the validityTime of the message is passed (see Table 1).

6.5.2.2 Source and destination

```
Type : PER
Sender : caching DANE
Receiver : DASH client
```

6.5.2.3 Data representation

Table 13 shows the DaneResourceStatus parameters.

Table 13 — DaneResourceStatus parameters

Parameter	Type	Cardinality	Description
DaneResourceStatus		1	Provides the status of the resources listed below.
status	enum	1	Specifies the resources that can be assigned to this type. The define types are documented in Table 14.
resource	anyURI	0 ... N	Provides a resource for which the status applies.
bytes	string	0 ... 1	If present, qualifies that the status only applies to the byte range provided in this attribute. The string shall conform to the byte-range-set according to RFC 7233:2014, 2.1. If not present, the status applies to the full resource.
resourceGroup	string	0 ... N	Provides a group of resources for which the status applies. In order to express different types of use cases and to compress the message size, some simplified regular expression patterns are permitted. The regular expression pattern follows the POSIX standard and only a subset of the meta character (see Table 15) shall be used such that the regular expression can only be used to define a finite number of segments. DANEs shall make sure that all segments identified by the resourceGroup regular expression have indeed the status conveyed in the message.

Table 14 — Allowed values for status parameter

Status	Semantics
cached	Resource is already cached in the DANE.
unavailable	Resource is not available in the DANE and request is expected to be responded with a 4xx code.
promised	Resource will be available in the DANE at the time announced in the Media Presentation.

Table 15 — Allowed meta character for resourceGroup regular expresion

Meta Character	Description
[ ]	A bracket expression. Matches a single character that is contained within the brackets. For example, [abc] matches “a”, “b”, or “c”. [a-z] specifies a range which matches any lowercase letter from “a” to “z”. These forms can be mixed: [abcx-z] matches “a”, “b”, “c”, “x”, “y”, or “z”, as does [a-cx-z].
( )	Defines a marked subexpression. The string matched within the parentheses can be recalled later (see the next entry, \n). A marked subexpression is also called a block or capturing group.
\n	Matches what the nth marked subexpression matched, where n is a digit from 1 to 9.
{m,n}	Matches the preceding element at least m and not more than n times. For example, a{3,5} matches only “aaa”, “aaaa”, and “aaaaa”.

6.5.3 SharedResourceAssignment

6.5.3.1 Motivation

This message allows the DANE to send to DASH clients competing for bandwidth over the same network information about how much bandwidth they should use in order to stay in a fair sharing of the total bandwidth.

This message is usually sent to DASH clients as a response to a SharedResourceAllocation message and is usually sent by a DANE who acts as a resource allocation entity.

NOTE The validity duration of this message is given by the validityTime parameter in the message envelope (see Table 1).

6.5.3.2 Source and destination

```
Type : PER
Sender : DANE
Receiver : DASH client
```

6.5.3.3 Data representation

Table 16 shows the SharedResourceAssignment parameters.

Table 16 — SharedResourceAssignment parameters

Parameter	Type	Cardinality	Description
SharedResourceAssignment	object	1	Response message from the coordinator that indicates the results of the bandwidth sharing operation.
clientId	token	1	The clientId identifies the target receiver of this message. This field shall use the same value as the senderId of the SharedResourceAllocation message sent by the client.
resourcePrice	double	0..N	A price for the bandwidth resource to be used by the receiver in its utility to price trade-off to determine an optimal operation point. The operation point selected will maximize the utility to price ratio.
bandwidth	int	0..1	This field contains the assigned bandwidth to the identified client. The unit is defined in bits/s. This field may be omitted if and only if resourcePrice information is present.

The validityTime attribute in the SAND message envelope shall be present for the SharedResourceAssignment message as it allows the DASH client to discover for how long the resource assignment is available.

6.5.4 MPDValidityEndTime

6.5.4.1 Motivation

This message provides the ability to signal to the client that a given MPD, whose @type is set to 'dynamic' and @minimumUpdatePeriod is present, can only be used up to at a certain wall-clock time.

Sending the message may be motivated by operational considerations such that DASH clients may fetch a new version of the MPD faster than they were planning to.

NOTE The validity duration of this message is given by the validityTime parameter in the message envelope (see Table 1).

6.5.4.2 Source and destination

```
Type : PER
Sender : DANE
Receiver : DASH Client
```

6.5.4.3 Data representation

Table 17 shows the MPDValidityEndTime parameters.

Table 17 — MPDValidityEndTime parameters

Parameter		Type	Cardinality	Description
MPDValidityEndTime		object	1	
	mpdId	string	0..1	The @id attribute of the corresponding MPD.
	publishTime	date-time	0..1	publish time attribute of the corresponding MPD with the same MPD@id.
	validityEndTime	date-time	1	Wall-clock time at which the MPD will no more be valid.
	mpdUrl	url	0..1	The recommended URL to use when fetching the next MPD update.
	mpd	string	0..1	The full updated MPD encoded in a string using base64 encoding.

mpdUrl and mpd parameters are mutually exclusive and one of the two shall be present in any MPDValidityEndTime message.

6.5.5 Throughput

6.5.5.1 Motivation

This message allows a DASH client to have, in advance, knowledge of the throughput characteristics and the guarantees along with this from the DANE to the DASH client. The status may be different for different baseUrl or different Representation IDs (repId) used, allowing to signal throughput characteristics dependent on the network delivering it. This may, for example, be used in case some QoS is provided on the access link between the DANE and the DASH client or if the data is cached in a local device. This message may apply in case the DANE is located in a different device from the DASH client, and communicate with the DASH client via a network, e.g. a mobile network.

The message expresses the throughput status of the network at the current time and for the duration in which the message is valid (see Table 1).

6.5.5.2 Source and destination

```
Type : PER
Sender : caching DANE
Receiver : DASH client
```


6.5.5.3 Data representation

Table 18 shows the throughput (with baseUrl) parameters while Table 19 shows the throughput (with representation ID) parameters.

Table 18 — Throughput (with baseUrl) parameters

Parameter	Type	Cardinality	Description
Throughput	object	1	
baseUrl	string	1	Provides the base URL for the associated resources, i.e. the throughput holds for all resources referenced by this base URL.
guaranteedThroughput	unsigned int	1	Specifies a guaranteed throughput in bits/s. Provides the guarantee for the throughput in a sense that the download time of a resource of size S bytes and from receiving the first byte to receiving the last byte is at most S*8 divided by the value of the attribute. This guarantee is provided with the below value of the percentage of certainty and holds for a request from the DASH client without any other concurrent HTTP requests.
percentage	unsigned int	0..1	Specifies the certainty of the above guarantee. The certainty of the guarantee is expressed as a percentage from 0 to 100. If not present, the default value is 100. Values higher than 100 are considered as 100.

Table 19 — Throughput (with representation ID) parameters

Parameter	Type	Cardinality	Description
Throughput	object	1	
repId	string	1	Provides the value for the representation ID, i.e. throughput holds for all resources associated to a Representation with the value of the parameter.
guaranteedThroughput	unsigned int	1	Specifies a guaranteed throughput in bit/s. Provides the guarantee for the throughput in a sense that the download time of a resource of size S bytes and from receiving the first byte to receiving the last byte is at most S*8 divided by the value of the attribute. This guarantee is provided with the below value of the percentage of certainty and holds for a request from the DASH client without any other concurrent HTTP requests.
percentage	unsigned int	0..1	Specifies the certainty of the above guarantee. The certainty of the guarantee is expressed as a percentage from 0 to 100. If not present, the default value is 100. Values higher than 100 are considered as 100.

6.5.6 AvailabilityTimeOffset

6.5.6.1 Motivation

This message allows a DASH client to have, in advance, knowledge of the availability time offset from the DANE to the DASH client. The status may be different for different baseUrl or different Representation IDs (repId) used, allowing to signal availability time offset dependent on the network delivering it. This is typically the result of the different paths and processing operations that the Segments of the Representation that is sent over a one network undergo compared to the segments of Representations that are delivered over another network. The intention is to adjust the segment availability start times to the new path and bring the DASH client to a correct operation point by taking into account the

delay caused by the transport of the resources over different networks. This availability offset may be positive or negative and should be taken into account by the DASH client to avoid buffer underflows when switching between Representations and also to avoid 404 messages as much as possible.

The message expresses the status of the network at the current time. The DANE should avoid significantly changing the parameters for one resource as it may result in scheduling/playback problems in the DASH client.

The segments'availability times shall be accurately signalled in the MPD. This message is not intended to be used as a general practice, but only on the exceptional occasions when the availability time of the segments is changed due to unforeseen conditions during the streaming session and after the publication of the MPD.

NOTE The validity duration of this message is given by the validityTime parameter in the message envelope (see Table 1).

6.5.6.2 Source and destination

```
Type : PER
Sender : caching DANE
Receiver : DASH client
```

6.5.6.3 Data representation

Table 20 shows the AvailabilityTimeOffset (with baseUrl) parameters while Table 21 shows the AvailabilityTimeOffset (with representation ID) parameters.

Table 20 — AvailabilityTimeOffset (with baseUrl) parameters

Parameter		Type	Cardinality	Description
AvailabilityTimeOffset		object	1	
	baseUrl	url	1	Provides the base URL for the associated resources, i.e. the offset holds for all resources referenced by this base URL.
	offset	int	1	Specifies the offset in milliseconds that needs to be applied to the segment availability start time of the resources accessible through the indicated location or representation identifier.

Table 21 — AvailabilityTimeOffset (with representation ID) parameters

Parameter		Type	Cardinality	Description
AvailabilityTimeOffset		object	1	
	repId	string	1	Provides the value for the representation ID, i.e. offset holds for all resources associated to a Representation with the value of the parameter.
	offset	int	1	Specifies the offset in milliseconds that needs to be applied to the segment availability start time of the resources accessible through the indicated location or representation identifier.

6.5.7 QoSInformation

6.5.7.1 Motivation

A DASH client can take the available network QoS information into consideration when requesting segments such that the consumed content bandwidth remains within the limits established by the

signalled QoS information. As such, there is a value in enabling signalling of QoS parameters to the DASH client in order to be used for adaptation purposes.

NOTE The validity duration of this message is given by the validityTime parameter in the message envelope (see Table 1).

6.5.7.2 Source and Destination

```
Type : PER
Sender : DANE
Receiver : DASH client
```

6.5.7.3 Data Representation

Table 22 shows the QoSInformation parameters.

Table 22 — QoSInformation parameters

Parameter		Type	Cardinality	Description
QoSInformation		object	1	
	gbr	int	0..1	Guaranteed bit rate in kbps between the DANE and DASH client, denoting the end-to-end guaranteed bit rate at the IP layer bearer that the service provider delivers to the DASH client.
	mbr	int	0..1	Maximum bit rate in kbps between the DANE and DASH client, limiting end to end bit rate that the service provider delivers the DASH client and denoting the end to end maximum bit rate at the IP layer bearer that the service provider delivers to the DASH client.
	delay	int	0..1	Packet layer budget in milliseconds (ms) denoting the maximum packet delay encountered at the IP layer with a confidence level of 98 percent.
	pl	int	0..1	Packet loss parameter, where packet loss rate equals 10^(-PL/10).

At least one of the above parameters shall exist in the message.

6.5.8 DeliveredAlternative

6.5.8.1 Motivation

As a response to an AcceptedAlternatives message sent by a DASH client, a DANE may deliver an alternative segment rather than the requested segment. If so, the DANE shall send a DeliveredAlternative message to the DASH client to inform him that the response contains a segment alternative and not the requested segment. The HTTP response shall prevent wrong additional caching from an intermediate RNE, thus it shall include the following headers.

— A `Warning` header with value '`214 Transformation Applied`'.

— A `Content-Location` header which value is the **`@sourceUrl`** of the selected segment representation.

— A `Vary` header containing the value '`SAND-AcceptedAlternatives`', in addition to other values that may be already present.

A DANE may also add no-caching directives into the response to prevent RNEs from caching the HTTP object.

NOTE The validity duration of this message is given by the validityTime parameter in the message envelope (see Table 1).

6.5.8.2 Source and Destination

```
Type : PER
Sender : DANE
Receiver : DASH client
```

6.5.8.3 Data Representation

Table 23 shows the DeliveredAlternative parameters.

Table 23 — DeliveredAlternative parameters

Parameter	Type	Cardinality	Description
DeliveredAlternative	object	1	Description of what is delivered when a DANE sends a response containing an alternative representation.
initialUrl	url	0..1	This is the URL of the initially requested segment. NOTE Within a request/response exchange in HTTP, the requested URL is implicitly known in the response because there is a 1 to 1 association between them. So it does not need to be repeated explicitly.
contentLocation	url	1	This is the URL of the actual delivered content.

6.5.9 DaneCapabilities

6.5.9.1 Motivation

This message allows a DANE to announce to a DASH client which specific SAND messages it supports (both as a receiver and emitter) by listing the MessageType (see Table 2) of all supported SAND messages. In addition, a message set URN that regroups several SAND messages together may also be used to signal additional support of a group of SAND messages.

NOTE The validity duration of this message is given by the validityTime parameter in the message envelope (see Table 1).

6.5.9.2 Source and destination

```
Type : PER
Sender : DANE
Receiver : DASH client
```

6.5.9.3 Data representation

Table 24 shows the DaneCapabilities parameters.

Table 24 — DaneCapabilities parameters

Parameter	Type	Cardinality	Description
DaneCapabilities	object	1	Provides the list of SAND messages supported by the DANE. A profile may also be used.
supportedMessage	array	0..N	Provides the list of SAND messages supported by the DANE.
messageType	int	1	MessageType of one supported SAND message (see Table 2).
messageSetUri	urn	0..1	URN of a message set that regroups several SAND messages together.

The only message set URN defined in the present specification is `urn:mpeg:dash:sand:messageset:all:2016`. DANEs which support this message set actually support all SAND messages defines in present specification.

6.6 PED Messages

This document does not include any PED message.

7 SAND message representation format

This clause defines XML as the format for SAND messages data representation format. While SAND network elements may implement additional data representation formats, they shall at least support SAND messages in XML format and HTTP header extension format (see 8.2.3).

The XML schema for SAND messages is defined in Annex A.

8 Transport Protocol to carry SAND messages

8.1 General

This clause defines HTTP as the minimum transport protocol that shall be at least supported by SAND enabled elements. It does not preclude that other additional transport protocols (as described in Clause 10) could also be implemented.

Delivery of the following messages shall be supported using HTTP protocol:

a) metrics messages (from DASH client to DANE);

b) status messages (from DASH client to DANE);

c) PER messages (from DANE to DASH client).

Depending on the nature of SAND messages, the use of HTTP protocol by SAND network elements varies. Table 25 summarizes which HTTP usages shall be supported (in bold in the table) by a SAND element or may be optional depending on the nature of the SAND message.

Table 25 — Mandatory usages of HTTP for carrying SAND messages

Metrics messages	**HTTP POST** (8.2.2) HTTP headers (8.2.3) may be used for small metrics messages.
Status messages	**HTTP headers** (8.2.3) HTTP POST (8.2.2) may be used when client knows DANE URL or IP address.
PER messages	**HTTP GET** (8.3)

A network element may implement other transport protocols in addition to HTTP or use HTTPS.

8.2 Protocol to carry metrics and status messages

8.2.1 General

Depending on how the destination DANE or Metrics server can be identified, two cases are defined. When the DANE URL or IP address is known by the DASH client, the client send messages as the body of HTTP requests directly sent to the DANE. When the target DANE(s) is (are) located on the media delivery path and not directly addressable as above, messages are attached to HTTP requests for media.

8.2.2 Sending a message directly to Metrics server or DANE

The DASH client uses the HTTP POST method to send a metrics (or status) message to the Metrics server (or the DANE). The URL for the request is the @endpoint URL announced by the channel signalling defined in Clause 9.

The `Content-Type` shall be `application/sand+xml`, as defined in Annex C.

The message document itself shall be included as the body part of the HTTP request.

For example, with @endpoint=http://some_dane.my_domain.com/path/to/messages the HTTP request is as follows.

```
POST /path/to/messages HTTP/1.1
Host: some_dane.my_domain.com
Content-Type: application/sand+xml;charset="utf-8"
Content-Length: 248
[...]

<?xml version="1.0" encoding="UTF-8"?>
<AnticipatedRequests messageId="1234" ...
```

8.2.3 Attaching a message to requests for media

The DANE to which a message is sent may not be the origin server but any intermediate network equipment on the media delivery path that is able to understand the HTTP request. In this case, the sending of SAND messages makes use of existing requests for DASH media, MPD or segments, from the DASH client to the DANE.

The DASH client may attach metrics or status messages to such requests by inserting in the request an HTTP header with name SAND-XXX where XXX is the name of the message as defined in Clause 6. Multiple headers may appear in a single request.

For SAND messages defined in another namespace than the MPEG namespace (urn:mpeg:dash:schema:sandmessage:2016), the namespace shall be present in the SAND message name with colon characters replaced with hyphens. For instance: SAND-urn-my-example-com-MyMessage.

The header value provides the message data. The header value shall conform to the following ABNF.

```
sand-message-value = sand-object
sand-object = sand-attr-or-list *( "," sand-attr-or-list )
sand-attr-or-list = sand-attribute / sand-list
sand-list = "[" sand-object *( ";" sand-object ) "]"
sand-attribute = sand-attribute-name "=" sand-value
sand-attribute-name = STRING
sand-value = QUOTEDSTRING / QUOTEDURI
/ TOKEN / INT / BYTERANGE / DATETIME
/ integer-list
integer-list = "[" INT *( "," INT ) "]"
```

`sand-attribute-name` shall be strings which contain the name of message parameters, as defined in this document.

`sand-value` shall contain a value that is allowed for the message parameter it is associated to, as defined in this document.

`integer-list` shall be used only for supportedMessage parameter of DaneCapabilities and ClientCapabilities messages.

QUOTEDURI is a URI (RFC 3986) enclosed between double quotes. If a double quote is part of the URI, it shall be encoded as %22. The enclosing double quotes protect from misinterpretation of delimiters used elsewhere in the message syntax.

DATETIME shall follow the ISO 8601 format, with a dot and not a comma for fractions of second to avoid confusion with comma separating key=value pairs. Additionally, the DATETIME format shall be

restricted to YYYYMMDDThhmmss.mmmZ format where the milliseconds part (.mmm) is optional. The simple "Z" at the end enforces the use of UTC (Zulu time) for time zone. The sender is assumed to know its own time zone and be able to produce this format easily.

If attributes defined for the envelope have to be included, they shall appear first in the message, and will be provided as key=value pairs. Note that the messageId parameter is not mandatory to be included in the case of status messages in HTTP headers.

When attaching messages to media request, messages are meant to be received by all DANEs on the media delivery path. Therefore, DANEs shall no strip header extensions so that other DANEs further in the delivery path may receive the message too.

NOTE If HTTPS is used for the media delivery, the message will only reach the final endpoint of the underlying TLS connection, and will not be seen by intermediates. Such SAND message communication can therefore not be used along with TLS if it is meant to be intercepted by an intermediate DANE.

The following text shows examples of status messages sent in HTTP headers.

```
SAND-AnticipatedRequests: [sourceUrl=~"http://my.cdn.com/video/some
_segment.m4v~",range=0-
5000,targetTime=2015-10-11T17:53:03Z]
SAND-SharedResourceAllocation: [bandwidth=300000,quality=1;bandwidth=6000
00,quality=2;bandwi
th=1200000,quality=3],weight=50,allocationStrategy=~"urn:mpeg:dash:sand:
allocation:basic:2016
~"
SAND-AcceptedAlternatives: [sourceUrl=~"/video/q_4/seg_25.mp4v~",range=0-
85333;sourceUrl=~"/vid
eo/q_3/seg_25.mp4v~",range=0-64000]
SAND-AbsoluteDeadline: deadline=2015-10-11T17:53:03Z
SAND-MaxRTT: maxRTT=2345
SAND-urn-my-example-com-MyMessage: foo=bar
SAND-ClientCapabilities: [messageType=6;messageType=7]
SAND-DaneCapabilities: supportedMessage=[13,14,16]
```

8.3 Protocol to carry PER messages

8.3.1 General

The following scenarios are considered for exchange between the DANE and the DASH client.

— Client assistance: A scenario for which the message is provided as auxiliary information for the client, but the service will be continued even if the client ignores the message. This is, for example, the case when the service provider provides information on the availability of additional networks that may be accessed by the DASH client to request the content. For examples of protocols and methods, see 8.3.2.

— Client Enforcement: A scenario for which the DASH client requires to act, the network provides suitable alternatives for future requests. The DANE cannot or is not willing to respond to the request with a valid resource, but provides suitable alternatives. For examples of protocols and methods, see 8.3.3.

— Error Cases: A scenario for which the client is informed that the request is not valid and the network provides the reason and possible resolutions for the problem. The DANE cannot respond to the request with a valid resource. For examples of protocols and methods, see 8.3.4.

8.3.2 Assistance

For assistance, a suitable method is the use of a dedicated HTTP header field that indicates a notification that the DANE has SAND messages to send to the DASH client. Upon receiving an HTTP entity that contains the SAND header field in its entity head, the DASH client issues a GET request to the indicated element to receive the SAND message.

The following ABNF syntax for the header field shall be used:

```
SAND-header-field = "MPEG-DASH-SAND" ":" element-address
element-address = absolute-URI
```

The field `absolute-URI` takes the syntax from RFC 3986. The SAND header field provides the URI to the SAND message that is to be fetched by the DASH client using an HTTP GET method.

8.3.3 Enforcement

For enforcement, a suitable method is the use of a 300 Multiple Choices response with the following details.

— The response includes an entity containing a SAND message. The entity format is specified by the media type given in the `Content-Type` and shall be set to `sand+xml`, as defined in Annex C.

— The response should not include the Location field to avoid the use of the Location field value by the user agent for automatic redirection.

This response is cacheable unless indicated otherwise.

8.3.4 Error case

For error cases, a suitable method is the use of a suitable 4xx error code. The response may include a SAND message from which the client can deduce the reason for the error code and potential resolution of the problem.

9 Signalling of SAND communication channel

9.1 General

The signalling mechanism described in this clause is intended to provide the necessary information so as to use HTTP protocol for SAND messages as described in Clause 8, or to signal the use of alternative transport protocols (such as WebSockets, see Clause 10).

In order to signal the SAND communication channel to DASH clients, the DANE may announce the presence of a SAND channel via the MPD using the **sand:Channel** element defined in the "`urn:mpeg:dash:schema:sand:2016`" namespace. The namespace prefix is "sand:". Table 26 shows the sand:Channel element.

Table 26 — sand:Channel element

Element or Attribute Name	Use	Description
Channel		provides information about a SAND channel
@id	O (string)	specifies an identifier for this SAND channel.
@schemeIdUri	M	identifies the channel scheme. The channel scheme defines the protocol the recipient of the SAND channel shall support.
@endpoint	O (string)	provides the endpoint to the SAND channel. The endpoint conforms to the URI specification, RFC 3986.
NOTE The conditions only holds without using xlink:href. If linking is used, then all attributes are "optional" and <minOccurs=0>		

The **sand:Channel**@schemeIdUri specifies which protocol the DASH client shall use with this SAND channel. Table 27 lists the mandatory protocols.

Table 27 — Protocols for SAND channels

@schemeIdURI	**Description**
urn:mpeg:dash:sand:channel:http:2016	The identifier indicates that the recipient of the SAND channel shall use the protocol defined in 8.2.2. That is, the SAND messages are transmitted via HTTP POST request inside the body of the request. In this case, the @endpoint of the **sand:Channel** shall be a valid HTTP URL according to RFC 3986.
urn:mpeg:dash:sand:channel:header:2016	The identifier indicates that the recipient of the SAND channel shall support the protocol defined in 8.2.3. That is, the SAND messages are transmitted via HTTP header extension of HTTP requests for media segments and MPD. In this case, the @endpoint of the **sand:Channel** shall not be present.

To allow signalling of a SAND communication channel in case it is not practical to modify the MPD, the DANE may also announce it via an HTTP header attached to any response to a media request.

The following ABNF syntax defines this header field:

```
SAND-channel-header-field = "MPEG-DASH-SANDChannel" ":" "schemeIdUri=" channel-scheme-name
"," optional-endpoint
optional-endpoint = "endpoint=" element-address / VOID
channel-scheme-name = URN
element-address = absolute-URI
```

The field absolute-URI follows the syntax from RFC 3986.

A DASH client implementing any SAND messages shall support signalling of SAND channel via both HTTP header and MPD.

9.2 XML schema for sand:Channel element

Figure 5 shows the XML schema for sand:Channel element.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="urn:mpeg:dash:schema:sand:2016"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:mpeg:dash:schema:sand:2016">

  <xs:annotation>
    <xs:appinfo>SAND elements</xs:appinfo>
    <xs:documentation xml:lang="en">
      This Schema defines the Server and network assisted DASH (SAND) XML elements for
MPEG-DASH.
    </xs:documentation>
  </xs:annotation>

  <!—SAND message: main element ->
  <xs:element name="Channel" type="ChannelType"/>

  <!—SAND Channel ->
  <xs:complexType name="ChannelType">
    <xs:attribute name="id" type="xs:string"/>
    <xs:attribute name="schemeIdUri" type="xs:anyURI" use="required"/>
    <xs:attribute name="endpoint" type="xs:anyURI" use="required"/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

</xs:schema>
```

Figure 5 — XML Schema for sand:Channel element

10 Optional transport protocols to carry SAND messages

10.1 General

This clause specifies additional transport protocols for the exchange of SAND messages between a DANE and the DASH client. Table 28 defines additional schemes that are optional protocols for SAND channels.

Table 28 — Optional protocols for SAND channels

`@schemeIdURI`	Description
`urn:mpeg:dash:sand:channel:websocket:2016`	The identifier indicates that the DASH client shall use the WebSocket Protocol as specified in 10.1 WebSocket Protocol. In this case, the `@endpoint` of the **`sand:Channel`** shall be a valid WebSocket URI as specified in 3 Web-Socket URIs of RFC 6455.

10.2 WebSocket protocol

10.2.1 General

This clause defines the exchange of SAND messages over the WebSocket Protocol as specified in RFC 6455.

10.2.2 Signalling via the MPD

This subclause addresses the signalling of the SAND channel setup information to the DASH clients via the MPD of which Figure 6 presents an MPD example.

The `@schemaIdUri` attribute of the **`sand:Channnel`** element shall be `urn:mpeg:dash:sand:channel:websocket:2016` while the `@endpoint` attribute shall contain a valid WebSoscket URI as defined in RFC 6455, Clause 3 WebSocket URIs.

The following is an MPD example of a SAND channel signalling.

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:mpeg:dash:schema:mpd:2011"
  xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd"
  xmlns:sand="urn:mpeg:dash:schema:sand:2016"
  type="dynamic"
  minimumUpdatePeriod="PT2S"
  timeShiftBufferDepth="PT30M"
  availabilityStartTime="2011-12-25T12:30:00"
  minBufferTime="PT4S"
  profiles="urn:mpeg:dash:profile:isoff-live:2011">

  <BaseURL>http://cdn1.example.com/</BaseURL>
  <BaseURL>http://cdn2.example.com/</BaseURL>

  <Period id="1">
    <!-- Video -->
    <AdaptationSet
      mimeType="video/mp4"
      codecs="avc1.4D401F"
      frameRate="30000/1001"
      segmentAlignment="true"
      startWithSAP="1">
      <BaseURL>video/</BaseURL>
      <SegmentTemplate timescale="90000" initialization="$Bandwidth%/init.mp4v"
media="$Bandwidth%/$Time$.mp4v">
        <SegmentTimeline>
          <S t="0" d="180180" r="432"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="v0" width="320" height="240" bandwidth="250000"/>
      <Representation id="v1" width="640" height="480" bandwidth="500000"/>
      <Representation id="v2" width="960" height="720" bandwidth="1000000"/>
    </AdaptationSet>
    <!-- Audio -->
    <AdaptationSet mimeType="audio/mp4" codecs="mp4a.40" lang="en" segmentAlignment="0"
startWithSAP="1">
      <SegmentTemplate timescale="48000" initialization="audio/en/init.mp4a"
media="audio/en/$Time$.mp4a">
        <SegmentTimeline>
          <S t="0" d="96000" r="432"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="a0" bandwidth="64000" />
    </AdaptationSet>
  </Period>

  <sand:Channel schemeIdUri="urn:mpeg:dash:sand:channel:websocket:2016"
endpoint="wss://cdn3.example.com">


</MPD>
```


Figure 6 — Example of setting up a Websocket channel as the SAND channel in the MPD

Upon reception of the MPD, the DASH client parses the MPD. The **sand:Channel** element indicates that a SAND channel is available. The @schemeIdUri attribute value specifies that the SAND channel uses the WebSocket Protocol. The @endpoint attribute indicates the actual endpoint to which the DASH client shall connect to in order to set up the SAND channel.

For the same security reasons as for the MPD delivery, it is recommended that WebSocket-based SAND channels run over TLS. Hence the use of "wss" for the URI scheme in the @endpoint attribute in the MPD example. See RFC 6455, 10.6 Connection confidentiality and integrity for more details.

When the connection to the WebSocket-based SAND channel is successful (see ISO/IEC 23009-1 for the successive steps of the protocol), the DASH client starts listening for incoming PER messages and may send metrics and status messages when needed. Since the WebSocket Protocol establishes a full-duplex connection, the DANE and the DASH client may exchange SAND messages travelling simultaneously in opposite directions over the channel.

10.2.3 WebSocket messages

Data frame messages of the WebSocket Protocol shall be set to the text type and the content shall be UTF-8 encoded as specified by the WebSocket Protocol. Each WebSocket message shall contain a valid SAND message compliant with the SAND message XML schema.

11 Reporting of metrics via SAND protocols

ISO/IEC 23009-1:2014, Annex D defines a list of metrics to be collected by the DASH client. SAND allows the MPD author to use the SAND protocol for the reporting of these metrics.

When SAND protocol is used to report metrics, the **Reporting**@schemeIdUri element shall indicate the SAND protocol that the DASH client shall use. The possible values are those listed in Table 27 and Table 28. In that case, the same rules applying to **sand:Channe**l@endpoint attribute shall apply to the **Reporting**@value attribute. In addition, the scheme given in Table 29 shall be used for the **Reporting**@schemeIdUri.

Table 29 — DASH metrics reporting

@schemeIdURI	**Description**
urn:mpeg:dash:sand:channel:2016	The identifier indicates that the DASH client shall use a SAND channel to report the metrics. In this case, the **Reporting**@value attribute shall contain the value of a **sand:Channe**l@id attribute present in the MPD. Consequently, the DASH client will derive the protocol to use from the attributes of the indicated **sand:Channel**.

Annex A
(normative)

XML Schema for SAND messages

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="urn:mpeg:dash:schema:sandmessage:2016"
  attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  xmlns:xs=http://www.w3.org/2001/XMLSchema
  xmlns="urn:mpeg:dash:schema:sandmessage:2016">

  <xs:annotation>
    <xs:appinfo>SAND Messages</xs:appinfo>
    <xs:documentation xml:lang="en">
      This Schema defines the Server And Network Assisted DASH (SAND) messages for MPEG-
DASH.
    </xs:documentation>
  </xs:annotation>

  <!-- SAND message: main element -->
  <xs:element name="SANDMessage" type="SANDEnvelopeType"/>

  <!-- SAND common envelope Type -->
  <xs:complexType name="SANDEnvelopeType">
    <xs:choice maxOccurs="unbounded">
      <xs:element name="AnticipatedRequests" type="AnticipatedRequestsType"/>
      <xs:element name="SharedResourceAllocation" type="SharedResourceAllocationType"/>
      <xs:element name="AcceptedAlternatives" type="AcceptedAlternativesType"/>
      <!-- AbsoluteDeadline is not allowed in XML, only in HTTP headers -->
      <xs:element name="MaxRTT" type="MaxRTTType"/>
      <xs:element name="NextAlternatives" type="NextAlternativesType"/>
      <xs:element name="ResourceStatus" type="ResourceStatusType"/>
      <xs:element name="DaneResourceStatus" type="DaneResourceStatusType"/>
      <xs:element name="SharedResourceAssignment" type="SharedResourceAssignmentType"/>
      <xs:element name="MPDValidityEndTime" type="MPDValidityEndTimeType"/>
      <xs:element name="Throughput" type="ThroughputType"/>
      <xs:element name="AvailabilityTimeOffset" type="AvailabilityTimeOffsetType"/>
      <xs:element name="QoSInformation" type="QoSInformationType"/>
      <!-- DeliveredAlternative is not allowed in XML, only in HTTP headers -->
      <xs:element name="DaneCapabilities" type="DaneCapabilitiesType"/>
      <!-- ISO/IEC 23009-5 Annex D DASH metrics -->
      <xs:element name="TcpList" type="TcpListType"/>
      <xs:element name="HttpList" type="HttpListType"/>
      <xs:element name="RepSwitchList" type="RepSwitchListType"/>
      <xs:element name="BufferLevelList" type="BufferLevelListType"/>
      <xs:element name="PlayList" type="PlayListType"/>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:choice>
    <xs:attribute name="senderId" type="xs:token”/>
    <xs:attribute name="generationTime" type="xs:dateTime”/>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:complexType>

  <!-- SAND message base Type -->
  <xs:complexType name="SANDMessageType">
    <xs:attribute name="messageId" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="validityTime" type="xs:dateTime”/>
  </xs:complexType>

  <!-- AnticipatedRequests Type -->
  <xs:complexType name="AnticipatedRequestsType">
    <xs:complexContent>
```

```
    <xs:extension base="SANDMessageType">
      <xs:sequence>
        <xs:element name="Request" type="AnticipatedRequestType" minOccurs="1"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Request Type -->
  <xs:complexType name="AnticipatedRequestType">
    <xs:attribute name="sourceUrl" type="xs:anyURI" use="required"/>
    <xs:attribute name="range" type="ByteRangeSetType"/>
    <xs:attribute name="targetTime" type="xs:unsignedLong”/>
  </xs:complexType>

  <!-- SharedResourceAllocation Type -->
  <xs:complexType name="SharedResourceAllocationType">
    <xs:complexContent>
    <xs:extension base="SANDMessageType">
      <xs:sequence>
        <xs:element name="OperationPoint" type="OperationPointType" minOccurs="1"
maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="weight" type="xs:unsignedInt”/>
      <xs:attribute name="allocationStrategy" type="xs:anyURI”/>
      <xs:attribute name="mpdUrl" type="xs:anyURI”/>
    </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- OperationPoint Type -->
  <xs:complexType name="OperationPointType">
    <xs:attribute name="bandwidth" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="quality" type="xs:unsignedInt”/>
    <xs:attribute name="minBufferTime" type="xs:unsignedInt”/>
  </xs:complexType>

  <!-- AcceptedAlternatives Type -->
  <xs:complexType name="AcceptedAlternativesType">
    <xs:complexContent>
    <xs:extension base="SANDMessageType">
      <xs:sequence>
        <xs:element name="Alternative" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="sourceUrl" type="xs:anyURI" use="required"/>
            <xs:attribute name="range" type="ByteRangeSetType"/>
            <xs:attribute name="bandwidth" type="xs:unsignedInt”/>
            <xs:attribute name="deliveryScope" type="xs:unsignedInt”/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- MaxRTT Type -->
  <xs:complexType name="MaxRTTType">
    <xs:complexContent>
    <xs:extension base="SANDMessageType">
      <xs:attribute name="maxRTT" type="xs:unsignedInt" use="required"/>
    </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- NextAlternatives Type -->
  <xs:complexType name="NextAlternativesType">
    <xs:complexContent>
    <xs:extension base="SANDMessageType">
      <xs:sequence>
        <xs:element name="Alternative" minOccurs="1" maxOccurs="unbounded">
```

```
          <xs:complexType>
            <xs:attribute name="sourceUrl" type="xs:anyURI" use="required"/>
            <xs:attribute name="range" type="ByteRangeSetType"/>
            <xs:attribute name="bandwidth" type="xs:unsignedInt”/>
            <xs:attribute name="deliveryScope" type="xs:unsignedInt”/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- ClientCapabilities Type -->
  <xs:complexType name="ClientCapabilitiesType">
    <xs:complexContent>
    <xs:extension base="SANDMessageType">
      <xs:sequence>
        <xs:element name="SupportedMessage" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="messageType" type="xs:unsignedInt" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="messageSetUri" type="xs:anyURI”/>
    </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- ResourceStatus Type -->
  <xs:complexType name="ResourceStatusType">
    <xs:complexContent>
      <xs:extension base="SANDMessageType">
     <xs:choice minOccurs="1" maxOccurs="unbounded">
           <xs:element name="ResourceURLInfo" type="ResourceURLInfoType"/>
       <xs:element name="ResourceRepresentationInfo"
type="ResourceRepresentationInfoType"/>
        </xs:choice>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- ResourceURLInfo Type -->
  <xs:complexType name="ResourceURLInfoType">
    <xs:attribute name="baseUrl" type="xs:anyURI”/>
    <xs:attribute name="status" type="ResourceStatusTypeStatusType" use="required"/>
    <xs:attribute name="reason" type="xs:string”/>
  </xs:complexType>

  <!-- ResourceRepresentationInfo Type -->
  <xs:complexType name="ResourceRepresentationInfoType">
    <xs:attribute name="repId" type="StringNoWhitespaceType"/>
    <xs:attribute name="status" type="ResourceStatusTypeStatusType" use="required"/>
    <xs:attribute name="reason" type="xs:string”/>

  </xs:complexType>

  <!-- ResourceStatus status enumeration -->
  <xs:simpleType name="ResourceStatusTypeStatusType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="available"/>
      <xs:enumeration value="cached"/>
      <xs:enumeration value="unavailable"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- DaneResourceStatus Type -->
  <xs:complexType name="DaneResourceStatusType">
    <xs:complexContent>
    <xs:extension base="SANDMessageType">
      <xs:sequence>
        <xs:element name="resource" type="ResourceType" minOccurs="0"
```

```
maxOccurs="unbounded"/>
        <xs:element name="resourceGroup" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="status" type="DaneResourceStatusTypeStatusType" use="required"/>
    </xs:extension>
    </xs:complexContent>
  </xs:complexType>

<!-- Resource Type -->
  <xs:complexType name="ResourceType">
    <xs:simpleContent>
      <xs:extension base="xs:anyURI">
        <xs:attribute name="bytes" type="xs:string”/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

<!-- DaneResourceStatus status enumeration -->
  <xs:simpleType name="DaneResourceStatusTypeStatusType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="cached"/>
      <xs:enumeration value="unavailable"/>
      <xs:enumeration value="promised"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- SharedResourceAssignment Type -->
  <xs:complexType name="SharedResourceAssignmentType">
    <xs:complexContent>
      <xs:extension base="SANDMessageType">
        <xs:sequence>
          <xs:element name="ResourcePrice" type="xs:decimal" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
      <xs:attribute name="clientId" type="xs:token" use="required"/>
      <xs:attribute name="bandwidth" type="xs:unsignedInt”/>
    </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- MPDValidityEndTime Type -->
  <xs:complexType name="MPDValidityEndTimeType">
    <xs:complexContent>
    <xs:extension base="SANDMessageType">
      <xs:sequence>
        <xs:choice>
          <xs:element name="MPDUrl" type="xs:anyURI”/>
          <xs:element name="MPD" type="xs:base64Binary”/>
        </xs:choice>
      </xs:sequence>
      <xs:attribute name="mpdId" type="xs:string”/>
      <xs:attribute name="publishTime" type="xs:dateTime”/>
      <xs:attribute name="validityEndTime" type="xs:dateTime" use="required"/>
    </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Throughput Type -->
  <xs:complexType name="ThroughputType">
    <xs:complexContent>
    <xs:extension base="SANDMessageType">
      <xs:attribute name="baseUrl" type="xs:anyURI”/>
      <xs:attribute name="repId" type="StringNoWhitespaceType"/>
      <xs:attribute name="guaranteedThroughput" type="xs:unsignedInt" use="required"/>
      <xs:attribute name="percentage" type="PercentageType"/>
    </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Percentage type -->
```

```
  <xs:simpleType name="PercentageType">
    <xs:restriction base="xs:unsignedInt">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="100"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- AvailabilityTimeOffset Type -->
  <xs:complexType name="AvailabilityTimeOffsetType">
    <xs:complexContent>
    <xs:extension base="SANDMessageType">
      <xs:attribute name="baseUrl" type="xs:anyURI”/>
      <xs:attribute name="repId" type="StringNoWhitespaceType"/>
      <xs:attribute name="offset" type="xs:unsignedInt" use="required"/>
    </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- QoSInformation Type -->
  <xs:complexType name="QoSInformationType">
    <xs:complexContent>
    <xs:extension base="SANDMessageType">
      <xs:attribute name="gbr" type="xs:unsignedInt”/>
      <xs:attribute name="mbr" type="xs:unsignedInt”/>
      <xs:attribute name="delay" type="xs:unsignedInt”/>
      <xs:attribute name="pl" type="xs:unsignedInt”/>
    </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- DaneCapabilities Type -->
  <xs:complexType name="DaneCapabilitiesType">
    <xs:complexContent>
      <xs:extension base="SANDMessageType">
    <xs:choice>
          <xs:sequence>
            <xs:element name="SupportedMessage" type="xs:unsignedInt" minOccurs="1"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:element name="MessageSetUri" type="xs:anyURI”/>
        </xs:choice>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>


  <!-- Metrics as defined in Annex D of ISO/IEC 23009-1 -->
  <!-- NOTE the naming convention complies with the keys defined in Annex D
       and with camelCase convention like the rest of the schema -->

  <!-- TcpList Type -->
  <xs:complexType name="TcpListType">
    <xs:complexContent>
      <xs:extension base="SANDMessageType">
        <xs:sequence>
          <xs:element name="TcpConnection" type="TcpConnectionType" minOccurs="1"
maxOccurs="unbounded" />
    </xs:sequence>
    </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- TcpConnection Type -->
  <xs:complexType name="TcpConnectionType">
      <xs:attribute name="tcpid" type="xs:unsignedInt" use="required"/>
      <xs:attribute name="dest" type="xs:string”/>
      <xs:attribute name="topen" type="xs:dateTime”/>
      <xs:attribute name="tclose" type="xs:dateTime”/>
      <xs:attribute name="tconnect" type="xs:unsignedInt”/>
  </xs:complexType>
```