IEC 62676-2-32

Edition 1.0 2019-06

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Video surveillance systems for use in security applications – Part 2-32: Recording control and replay based on web services

Systèmes de vidéosurveillance destinés à être utilisés dans les applications de sécurité – Partie 2-32: Contrôle d'enregistrement et lecture en fonction des services Web

IEC 62676-2-32:2019-06(en-fr)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

IEC 62676-2-32

Edition 1.0 2019-06

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Video surveillance systems for use in security applications – Part 2-32: Recording control and replay based on web services

Systèmes de vidéosurveillance destinés à être utilisés dans les applications de sécurité – Partie 2-32: Contrôle d'enregistrement et lecture en fonction des services Web

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 13.320

ISBN 978-2-8322-7036-3

CONTENTS

FOREWORD 6
INTRODUCTION 8
1 Scope 9
2 Normative references 9
3 Terms, definitions and abbreviated terms 10
3.1 Terms and definitions 10
3.2 Abbreviated terms 11
4 Overview 11
4.1 Interfaces 11
4.2 Storage model 12
4.3 Recording control 13
4.4 Search 14
4.5 Replay control 14
4.6 Export file format 14
4.6.1 Layout 14
4.6.2 Use case 1: Playback of chunked and oversize clips at remote site 15
4.6.3 Use case 2: Forensic analysis at court 16
4.6.4 Use case 3: Playback at players not equipped according to the present specification 16
4.7 Receiver 16
5 Recording control service 16
5.1 Overview 16
5.2 General requirements 18
5.3 Data structures 18
5.3.1 RecordingConfiguration 18
5.3.2 TrackConfiguration 18
5.3.3 RecordingJobConfiguration 18
5.4 CreateRecording 20
5.5 DeleteRecording 21
5.6 GetRecordings 21
5.7 SetRecordingConfiguration 22
5.8 GetRecordingConfiguration 22
5.9 CreateTrack 23
5.10 DeleteTrack 24
5.11 GetTrackConfiguration 24
5.12 SetTrackConfiguration 25
5.13 CreateRecordingJob 25
5.14 DeleteRecordingJob 26
5.15 GetRecordingJobs 27
5.16 SetRecordingJobConfiguration 27
5.17 GetRecordingJobConfiguration 28
5.18 SetRecordingJobMode 28
5.19 GetRecordingJobState 29
5.20 GetRecordingOptions 31
5.21 ExportRecordedData 31
5.22 StopExportRecordedData 32
5.23 GetExportRecordedDataState 33

5.24 GetServiceCapabilities 34
5.25 Events 35
5.25.1 General 35
5.25.2 Recording job state changes 35
5.25.3 Configuration changes 35
5.25.4 Data deletion 36
5.25.5 Recording and track creation and deletion 36
5.26 Examples 37
5.26.1 Example 1: setup recording of a single camera 37
5.26.2 Example 2: Record multiple streams from one camera to a single recording 38
6 Search service 38
6.1 General 38
6.2 Concepts 39
6.2.1 Search direction 39
6.2.2 Recording event 39
6.2.3 Search session 39
6.2.4 Search scope 40
6.2.5 Search filters 40
6.2.6 Time information 40
6.3 Data structures 40
6.3.1 RecordingInformation structure 40
6.3.2 RecordingSourceInformation structure 41
6.3.3 TrackInformation structure 41
6.3.4 SearchState Enumeration 42
6.3.5 MediaAttributes structure 42
6.3.6 FindEventResult structure 42
6.3.7 FindPTZPositionResult structure 42
6.3.8 PTZPositionFilter structure 42
6.3.9 MetadataFilter structure 43
6.3.10 FindMetadataResult structure 43
6.4 GetRecordingSummary 43
6.5 GetRecordingInformation 43
6.6 GetMediaAttributes 44
6.7 FindRecordings 45
6.8 GetRecordingSearchResults 45
6.9 FindEvents 46
6.10 GetEventSearchResults 47
6.11 FindPTZPosition 48
6.12 GetPTZPositionSearchResults 49
6.13 FindMetadata 50
6.14 GetMetadataSearchResults 51
6.15 EndSearch 52
6.16 GetServiceCapabilities 53
6.17 Recording event descriptions 53
6.18 XPath dialect 54
7 Replay control 55
7.1 Request replay URI 55
7.2 ReplayConfiguration 56

7.3 SetReplayConfiguration 56
7.4 GetReplayConfiguration 56
7.5 GetServiceCapabilities 57
8 Playback 57
8.1 RTSP Usage 57
8.2 RTSP describe 58
8.3 RTP header extension 58
8.3.1 General 58
8.3.2 NTP timestamps 59
8.3.3 Compatibility with the JPEG header extension 59
8.4 RTSP feature tag 60
8.5 Initiating playback 60
8.5.1 General 60
8.5.2 Range header field 60
8.5.3 Rate-Control header field 61
8.5.4 Frames header field 61
8.5.5 Synchronization points 62
8.6 Reverse replay 62
8.6.1 Initiation 62
8.6.2 Packet transmission order 62
8.6.3 RTP sequence numbers 64
8.6.4 RTP timestamps 64
8.7 RTSP Keepalive 65
8.8 Currently recording footage 65
8.9 End of footage 65
8.10 Go To Time 65
8.11 Use of RTCP 65
9 Export file format 66
9.1 Required side information 66
9.2 Timing 68
9.3 Correction of start time 68
9.4 Signature 68
9.4.1 Preparing the signature input 68
9.4.2 Generating the signature 68
9.4.3 Include the generated signature in the file 69
9.5 Repeated signing 70
10 Receiver service 71
10.1 General 71
10.2 Synchronization points 72
10.3 Persistence 72
10.4 Receiver modes 72
10.5 Receiver commands 72
10.5.1 GetReceivers 72
10.5.2 GetReceiver 73
10.5.3 CreateReceiver 73
10.5.4 DeleteReceiver 73
10.5.5 ConfigureReceiver 74
10.5.6 SetReceiverMode 74
10.5.7 GetReceiverState 75

10.6 GetServiceCapabilitites 75
10.7 Events 76
10.7.1 General 76
10.7.2 ChangeState 76
10.7.3 Connection Failed 76
Annex A (informative) Repeated signing 77
Annex B (normative) Schema files 79
B.1 Recording control 79
B.2 Search 89
B.3 Replay control 96
B.4 Receiver 98
B.5 Common Schema 102
Bibliography 110

Figure 1 – Storage model with tracks 13
Figure 2 – Sealing and examination in a nutshell (Source: Wikipedia) 15
Figure 3 – Example of recordings and tracks 17
Figure 4 – RecordingJobConfiguration structure 19
Figure 5 – RecordingJobStateInformation structure 30
Figure 6 – Recording state chart 41
Figure 7 – Packet transmission during forward playback 63
Figure 8 – Packet transmission during reverse playback 64
Figure A.1 – Single signature box arrangement 77
Figure A.2 – Repeated signature box arrangement 77

Table 1 – Referenced namespaces (with prefix) 12
Table 2 – Track configuration 21
Table 3 – RTP packet layout 58
Table 4 – RTP packet with JPEG header layout 59

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

VIDEO SURVEILLANCE SYSTEMS FOR USE IN SECURITY APPLICATIONS –

Part 2-32: Recording control and replay based on web services

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 62676-2-32 has been prepared by IEC technical committee 79: Alarm and electronic security systems.

This first edition, together with IEC 60839-11-31 and IEC 62676-2-31, cancels and replaces IEC 62676-2-3:2013.

This edition includes the following significant technical changes with respect to IEC 62676-2-3:2013:

a) an export file format has been added.

The text of this International Standard is based on the following documents:

FDIS	Report on voting
79/621/FDIS	79/623/RVD

Full information on the voting for the approval of this International Standard can be found in the report on voting indicated in the above table.

This document has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all parts in the IEC 62676 series, published under the general title *Video surveillance systems for use in security applications*, can be found on the IEC website.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

The goal of this document is to provide a fully interoperable network video recording and reply implementation comprised of products from different vendors. This document describes the network video recording model, interfaces, data types and data exchange patterns. The document reuses existing relevant standards where available, and introduces new specifications only where necessary to support the specific requirements for network video recording and reply.

VIDEO SURVEILLANCE SYSTEMS FOR USE IN SECURITY APPLICATIONS –

Part 2-32: Recording control and replay based on web services

1 Scope

This part of IEC 62676 specifies the web service interface for the configuration of the recording of video, audio and metadata. Additionally, associated events are defined.

Clause 4 provides a definition of the storage model this document is based on.

Web service usage is outside the scope of this document. Please refer to the IEC 60839-11-31 for more information

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.

IEC 60839-11-31:2016, *Alarm and electronic security systems – Part 11-31: Electronic access control systems – Core interoperability protocol based on Web Services*

IEC 62676-2-31:2019, *Video surveillance system for use in security applications – Part 2-31: Live streaming and control based on web services*

Internet Assigned Numbers Authority (IANA), Media Types, *Media Types* [online]. Edited N. Freed et al. [viewed 2019-02-28]. Available at https://www.iana.org/assignments/media-types/media-types.xhtml

INTERNET ENGINEERING TASK FORCE (IETF). RFC 2326: *Real Time Streaming Protocol (RTSP)* [online]. Edited by H. Schulzrinne et al. April 1998 [viewed 2019-02-28]. Available at http://www.ietf.org/rfc/rfc2326.txt

INTERNET ENGINEERING TASK FORCE (IETF). RFC 3280, *Internet X.509 Public Key Infrastructure – Certificate and Certificate Revocation List (CRL) Profile* [online]. Edited by Housley, et. al. April 2002 [Viewed 2019-02-28]. Available at http://www.ietf.org/rfc/rfc3280.txt

INTERNET ENGINEERING TASK FORCE (IETF). RFC 3550, *RTP: A Transport Protocol for Real-Time* [online]. Edited by Schulzrinne, et al. Jul 2003 [viewed 2019-02-28]. Available at https://www.ietf.org/rfc/rfc3550.txt

INTERNET ENGINEERING TASK FORCE (IETF). RFC 4055, *Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure – Certificate and Certificate Revocation List (CRL)* Profile [online]. Edited by Schaad, et al. June 2005 [viewed 2019-02-28]. Available at https://www.ietf.org/rfc/rfc4055.txt

The World Wide Web Consortium (W3C). SOAP12-PART1, SOAP 1.2 – Part 1, Messaging Framework [online]. Edited by M, Gudgin et al. Apr 2007 {Viewed 2019-02-28]. Available at https://www.w3.org/TR/soap12-part1/

The World Wide Web Consortium (W3C). XML-Schema 1, *W3C XML Schema – Part 1: Structures Second Edition* [online]. Edited by H. Thompson et al. Oct 2004 [viewed 2019-02-28]. Available at https://www.w3.org/TR/xmlschema-1/

The World Wide Web Consortium (W3C). XML-Schema 2, *W3C XML Schema – Part 2: Datatypes Second Edition* [online]. Edited by P. Biron et al. Oct 2004 [viewed 2019-02-28]. Available at https://www.w3.org/TR/xmlschema-2/

The World Wide Web Consortium (W3C). XPath 1.0, *XML Path Language (XPath) Version 1.0* [online]. Edited by J, Clark et al. Nov 1999 [Viewed 2019-02-28]. Available at https://www.w3.org/TR/1999/REC-xpath-19991116/

Federal Information Processing Standard (FIPS), FIPS 180-4, *Secure Hash Standard (SHS)* [online]. [viewed 2019-02-28]
Available at https://csrc.nist.gov/publications/detail/fips/180/4/final

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions 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
access unit
one or more frames or samples of audio, video, or metadata, which are contained in a group of RTP packets having the same presentation time

3.1.2
certificate
data which binds a public key to a subject entity

Note 1 to entry: The certificate is digitally signed by the certificate issuer to allow for verifying its authenticity.

3.1.3
metadata
streaming data except video and audio, including video analytics results, PTZ position data and other metadata (such as textual data from POS applications)

3.1.4
recording
container for a set of audio, video and metadata tracks

Note 1 to entry: A recording can hold one or more tracks. A track is viewed as an infinite timeline that holds data at certain times.

3.1.5
recording event
event associated with a recording, represented by a notification message in the APIs

3.1.6
recording job
job that performs the transfer of data from a data source to a particular recording using a particular configuration

3.1.7
signature
digital signature
digital signature scheme
mathematical scheme for demonstrating the authenticity of a digital message or document

3.1.8
track
individual data channel consisting of video, audio, or metadata

Note 1 to entry: This definition is consistent with the definition of "track" in IETF RFC 2326.

3.1.9
video analytics
algorithms or programs used to analyse video data and to generate data describing object location and behaviour

3.2 Abbreviated terms

CCTV	closed-circuit television
JPEG	Joint Photographic Expert Group
PTZ	pan tilt zoom
RTCP	RTP control protocol
RTP	real-time transport protocol
RTSP	real time streaming protocol
SDP	session description protocol
SHA	secure hashing algorithm
TCP	transmission control protocol
UDP	user datagram protocol
UTC	coordinated universal time
UTF	Unicode Transformation Format
WSDL	web service description language

4 Overview

4.1 Interfaces

This document provides a set of interfaces that enable the support of interoperable network storage devices, such as network video recorders, digital video recorders and cameras with embedded storage.

The following functions are supported:

- recording control;
- search;
- replay control.

These functions are provided by three interrelated services:

Recording control service enables a client to manage recordings, and to configure the transfer of data from data sources to recordings. Managing recordings includes creation and deletion of recordings and tracks. The WSDL for this service is specified in Clause B.1

Search service enables a client to find information about the recordings on the storage device, for example to construct a "timeline" view, and to find data of interest within a set of recordings. The latter is achieved by searching for events that are included in the metadata track recording. The WSDL for this service is specified in Clause B.2

Replay control service enables a client to play back recorded data, including video, audio and metadata. Functions are provided to start and stop playback and to change speed and direction of the replayed stream. It also enables a client to download data from the storage device so that export functionality can be provided. The WSDL for this service is specified in Clause B.3

This document also includes specification for:

- playback of recorded data;
- export file format;
- a receiver that acts as a RTSP client endpoint. The WSDL for this service is specified in Clause B.4.

Table 1 lists the prefix and namespaces used in this specification. For interfaces defined by this document, the respective annex is provided. Listed prefixes are not part of this document and an implementation can use any prefix.

Table 1 – Referenced namespaces (with prefix)

Prefix	Namespace URI	Reference
env	http://www.w3.org/2003/05/soap-envelope	*W3C SOAP12-PART1*
ter	http://www.onvif.org/ver10/error	IEC 60839-11-31
trc	http://www.onvif.org/ver10/recording/wsdl	Clause B.1
trp	http://www.onvif.org/ver10/replay/wsdl	Clause B.3
trv	http://www.onvif.org/ver10/receiver/wsdl	Clause B.4
tse	http://www.onvif.org/ver10/search/wsdl	Clause B.2
tt	http://www.onvif.org/ver10/schema	Clause B.5
xs	http://www.w3.org/2001/XMLSchema	W3C XML-Schema-1 W3C XML-Schema-2

4.2 Storage model

The storage interfaces in this document present a logical view of the data on the storage device. This view is completely independent of the way data might be physically stored on the disk.

The key concept in the storage model is that of a recording. The term recording is used in this specification to denote a container for a set of related audio, video and metadata tracks, typically from the same data source, for example a camera. A recording could hold any number of tracks. A track is viewed as an infinite timeline that holds data at certain times.

At a minimum, a recording is capable of holding three tracks, one for audio, one for video and one for metadata. Some implementations of the recording service can support multiple tracks of each type. For example, the same recording could hold two video tracks, one containing a low-resolution or low-frame-rate stream and one containing a high-resolution or high-frame-rate stream; see Figure 1.

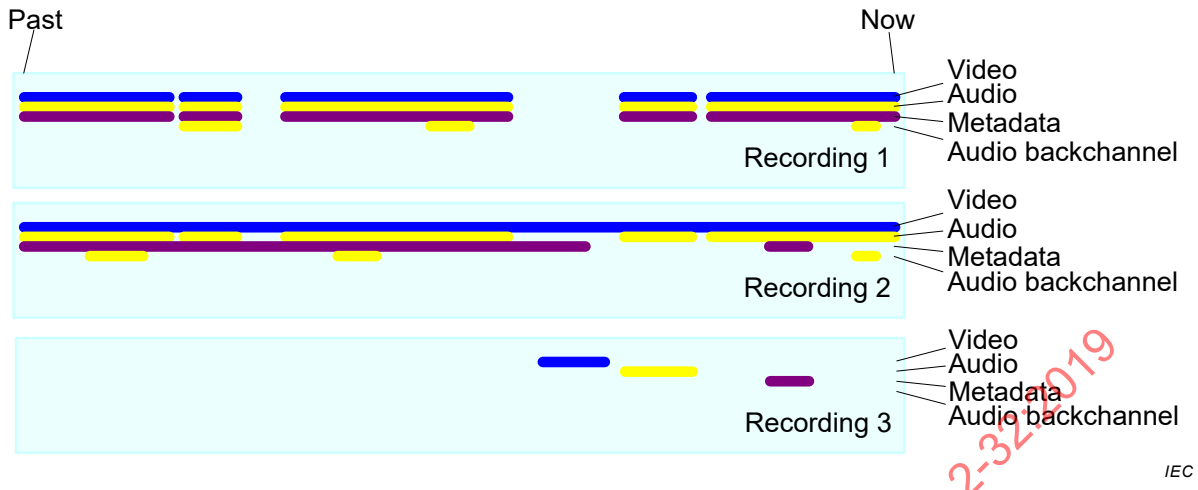

Figure 1 – Storage model with tracks

It is important to note that the storage interfaces do not expose the internal storage structures on the device. In particular, a recording is not intended to represent a single file on disk, although, in many storage device implementations, a recording is physically stored in a series of files. For instance, some camera implementations realise alarm recording by creating a distinct file for each alarm that occurs. Although each file could be represented as a different recording, the intent of the model in this document is that all these files are aggregated into a single recording.

Within a recording the regions where data is actually recorded are represented by pairs of events, where each pair comprises an event when recording started and an event when recording stopped. A client can construct the logical view of the recordings by using the FindRecordings and FindEvents methods of the search service.

If metadata is recorded, the metadata track can hold all the events generated by the data source, see Clause 10 of IEC 60839-11-31:2016 and 6.3.8 of IEC 62676-2-31:2019. In addition, a device also conceptually records historical events as defined by this document (see 6.17). This includes information such as the start and end of a recorded data range. A device can also conceptually record vendor specific historical events. Events generated by the device are not inserted in existing metadata tracks of recordings. The FindEvents method in the search service can find all the recorded events.

4.3 Recording control

The recording service enables a client to manage recordings, and to configure the transfer of data from data sources to recordings. Managing recordings includes the creation and deletion of recordings and tracks.

Recording jobs transfer data from a recording source to a recording. A recording source can be a receiver object created with the receiver service, or it can be a media profile that encodes data on a local device. The media profile could be used as a source on a camera with embedded storage.

To save data to a recording, a client first creates a recording and ensures that the recording has the necessary tracks. Then the client creates a recording job that pulls data from one or more sources and stores the data to the tracks in the recording.

Clients may set up multiple recording jobs that all record into the same recording. If multiple recording jobs are active, the device uses a priority scheme to select between the tracks defined in the recording jobs. Clients may change the mode of recording jobs at any time, thereby providing means to implement features like alarm recording or manual recording.

The recording job relies on the receiver service for receiving the data from other devices through receiver objects identified by ReceiverTokens.

4.4 Search

The search service enables a client to find information about the recordings on the storage device, for example to construct a "timeline" view, and to find data of interest within a set of recordings. The latter is achieved by searching for events and other information that is included in the metadata track recording.

The search service provides the following functionality:

- find recordings and information about each recording;
- find events in the metadata and among the historical events;
- find PTZ positions in the metadata;
- find other information in the metadata e.g. text from EPOS (electronic point-of-sale) systems.

The actual searching is done by coupled find and result operations and is asynchronous. Each find operation initiates a search session. The client can then acquire the results from the search session in increments, or all at once, depending on implementation and the scale of the search. There are four pairs of search operations for recordings, recording events, PTZ positions and metadata:

- FindRecordings and GetRecordingSearchResults;
- FindEvents and GetEventSearchResults;
- FindPTZPosition and GetPTZPositionSearchResults;
- FindMetadata and GetMetadataSearchResults.

4.5 Replay control

The replay control service provides a mechanism for the replay of stored video, audio and metadata. This mechanism can also be used to download data from the storage device so that export functionality can be provided.

The replay protocol is based on RTSP (IETF RFC 2326). However because RTSP does not directly support all of the requirements for replay, several extensions have been added to the protocol. In particular, an RTP header extension is defined to allow an absolute timestamp to be associated with each access unit (e.g. video frame), and to convey information about stream continuity.

The GetReplayUri command in the replay service returns the RTSP URL of a recording to allow it to be replayed using RTSP.

4.6 Export file format

4.6.1 Layout

All data a user wishes to carry away separately are put into a metaphorical bag. The bag is than sealed to enable tamper detection. Anyone wanting to use the data from the bag first examines the seal. The data in the bag are identical with the original data as long as the seal is intact, see Figure 2.

Here, the metaphorical bag is represented by a file and the seal is represented by a signature over all data in the file.

The "bag of evidence" approach builds on procedures for media data and related metadata to be securely extracted from a trusted storage in a separate file. This approach defines which metadata have to be preserved in order to provide accurate replay. Data are provided "as is" without any further assertions, whatsoever, to perpetuate evidence.

Processing power reliefs include usage of hash functions before signature algorithms are applied. Multiple stages of signatures might be applied to collect additional information into a single sealed file.

International state of the art standards are applied for the file structure, hash and signature algorithms. The surveillance application format defined in ISO/IEC 23000-10 and the RSA2048 signature as well as the SHA-256 hash algorithm approved by NIST come into operation for the most widespread interoperability.

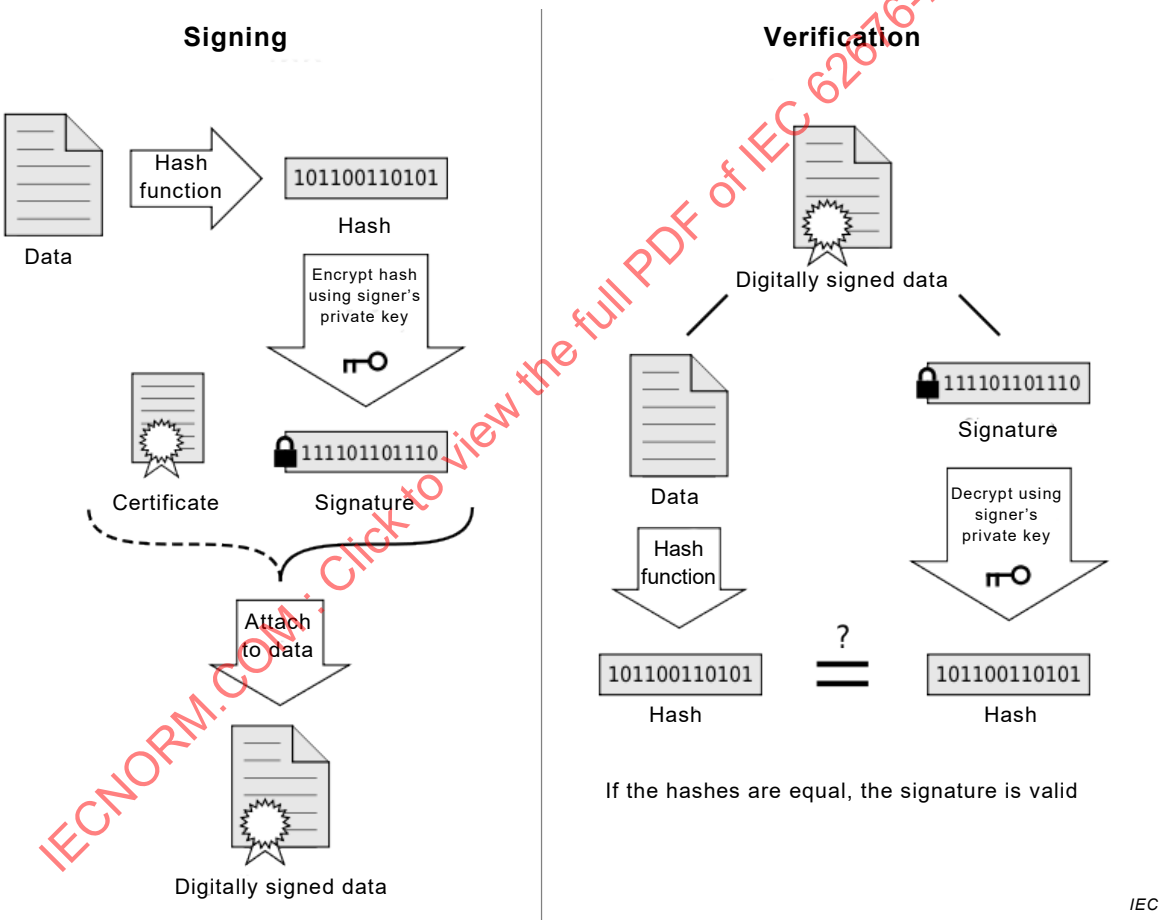

Figure 2 – Sealing and examination in a nutshell (Source: Wikipedia)

4.6.2 Use case 1: Playback of chunked and oversize clips at remote site

An operator exports a video clip with associated audio from a DVR of brand A onto two DVDs, because it didn't fit on one. The selected recording period contains gaps because the recorder only recorded when motion was detected. The DVD is then sent to a second site with software where the content of the DVDs is copied to the local hard disk. The user then plays it back in the video management system of brand B. The operator at the playback station wants to see the gaps in the recording, and seek any time where video has been exported. On playback, he expects the video to playback smoothly with lip sync audio.

4.6.3 Use case 2: Forensic analysis at court

A court receives video clips from a grocery store, a street surveillance system and a metro operator. All three videos are shown in the court's approved video player.

The judges want to see the suspect in all three video clips with exact time information. They also want to have information when the video clips have been exported and whether the video sequence is complete and authentic.

4.6.4 Use case 3: Playback at players not equipped according to the present specification

An authorized person receives video clips in the format defined in the present specification and wants to play back the media data on players conforming to the underlying standard definitions. Interpretation of the additional information added by the present specification is not required

4.7 Receiver

A receiver is an object that acts as an RTSP client endpoint. Receivers are used by other services that consume media streams, such as the recording and services. A receiver has a configuration that determines the RTSP endpoint to which it should connect and the connection parameters it should use.

A receiver can operate in three distinct modes:

- AlwaysConnect: the receiver attempts to maintain a persistent connection to the configured endpoint.
- NeverConnect: the receiver does not attempt to connect.
- AutoConnect: the receiver connects on demand, as required by consumers of the media streams.

A single receiver may be used by more than one consumer. For example, in order to record a stream and also perform analytics on it, both a recording job and an analytics engine could be attached to the same receiver. If the receiver uses the "Auto Connect" mode, it will connect whenever either the recording job or the analytics engine is active, and disconnect when neither of them are active.

Receivers may be created and deleted either manually, by calling the CreateReceiver and DeleteReceiver operations in the Receiver Service, or automatically by other services. For example, if a recording job is created with the "AutoCreateReceiver" option, it will automatically create and attach to a receiver. Deleting the recording job will also delete the receiver.

5 Recording control service

5.1 Overview

The recording service enables a client to manage recordings, and to configure the transfer of data from data sources to recordings. Managing recordings includes creation and deletion of recordings and tracks, as well as locking and unlocking ranges of recordings and deletion of recorded data.

Recording jobs transfer data from a recording source to a recording. A recording source can be a receiver object created with the receiver service, or it can be a media profile that encodes data on a local device. The media profile could be used as a source on a camera with embedded storage.

The term "recording" is used in this specification to denote a container for a set of audio, video and metadata tracks. A recording could hold any number of tracks. A track is viewed as an infinite timeline that holds data at certain times.

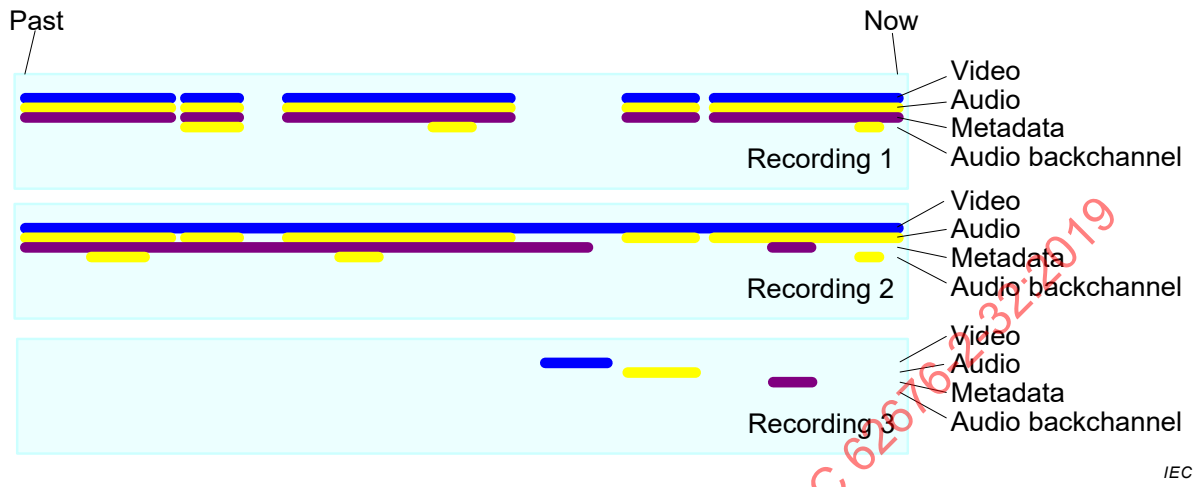

Figure 3 – Example of recordings and tracks

Figure 3 shows three recordings, each with a video, a metadata and two audio tracks. Here, a second audio track is used for storing the audio backchannel.

As a minimum, a recording shall be capable of holding three tracks: one for audio, one for video and one for metadata. Some implementations of the recording service can support multiple tracks of each type. All recorded data of a track shall have the same encoding.

To save data to a recording, a client first creates a recording and ensures that the recording has the necessary tracks. Then the client creates a recording job that pulls data from one or more sources and stores the data to the tracks in the recording.

Clients may set up multiple recording jobs that all record into the same recording. If multiple recording jobs are active, the device uses a priority scheme to select between the tracks defined in the recording jobs. Clients may change the mode of recording jobs at any time, thereby providing means to implement features such as alarm recording or manual recording.

The recording job relies on the receiver service for receiving the data from other devices through receiver objects identified by ReceiverTokens.

For the cases where a client uses a receiver object with a single recording job, the recording service can auto create and auto delete receiver objects. Autocreation is signalled with the AutoCreateReceiver flag in the recording job configuration structure. Receiver objects created this way shall be automatically deleted when no recording job uses them anymore. A receiver object that is automatically created shall have all its fields set to empty values. The client should configure the receiver object after it has created the recording job.

This document's view of recordings is a logical one, which is independent of the way recordings are physically stored on disk. For instance, some camera implementations realise alarm recording by creating a distinct file on a FAT file system for each alarm that occurs. Although each file could be represented as a different recording as defined by this document, the intent of the model in this document is that all these files are aggregated into a single recording. By searching for the "DataPresent" event with the FindEvents method of the search service, a client can locate the times at which video started to be recorded and where video stopped being recorded.

If metadata is recorded, the metadata track can hold all the events generated by the data source, see Clause 10 of IEC 60839-11-31:2016 and 6.3.8 of IEC 62676-2-31:2019. In addition, a device also conceptually record historical events as defined by this document, see 6.17, this includes information such as the start and end of a recorded data range. A device may also conceptually record vendor-specific historical events. Events generated by the device are not inserted in existing metadata tracks of recordings. The FindEvents method in the search service can find all the recorded events.

Many device implementations will automatically delete the oldest recorded data from storage in order to free up space for new recordings. Locks provide a mechanism to allow a user to select ranges of data. A range of data that is locked does not get deleted automatically. Support for locks is reserved for future versions of the specification.

5.2 General requirements

All the objects created within the recording service shall be persistent – i.e. they shall survive a power cycle. Likewise, all the configuration data in the objects shall be persistent.

5.3 Data structures

5.3.1 RecordingConfiguration

The RecordingConfiguration structure shall be used to configure recordings through CreateRecordings and Get/SetRecordingConfiguration.

MaximumRetentionTime specifies the maximum time that data in any track within the recording shall be stored. The device shall delete any data older than the maximum retention time. Such data shall not be accessible anymore. If the MaximumRetentionPeriod is set to 0, the device shall not limit the retention time of stored data, except by resource constraints. Whatever the value of MaximumRetentionTime, the device may automatically delete recordings to free up storage space for new recordings.

None of the other fields defined in this structure shall be used by the device. Instead, it simply stores this information, and it shall return it through the GetRecordingConfiguration and GetRecordingInformation methods.

A device may truncate any descriptive string without causing a fault if it exceeds the supported length. Descriptive strings are Location, Description and Content.

5.3.2 TrackConfiguration

The TrackConfiguration structure shall be used to configure tracks using CreateTrack and Get/SetTrackConfiguration.

TrackConfiguration contains the following fields:

TrackType defines the data type of the track. It shall be equal to the strings "Video", "Audio" or "Metadata". The track shall only be able to hold data of that type.

None of the other fields defined in this structure shall be used by the device. Instead, it simply stores this information, and it shall return it through the GetTrackConfiguration and GetRecordingInformation methods.

5.3.3 RecordingJobConfiguration

The RecordingJobConfiguration structure shall hold the configuration for a recording job. As a UML diagram, the RecordingJobConfiguration can be viewed as shown in Figure 4

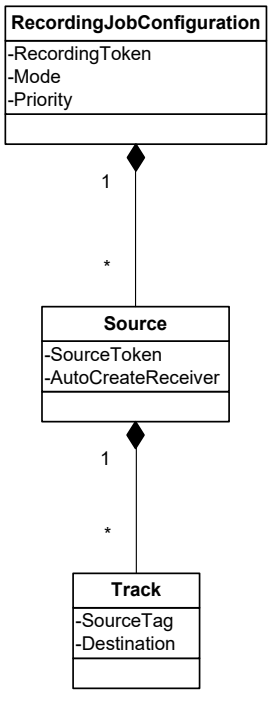

Figure 4 – RecordingJobConfiguration structure

The RecordingJobConfiguration holds the following fields:

RecordingToken: Identifies the recording to which this job shall store the received data.

Mode: If it is idle, nothing shall happen. If it is active and the recording job has the highest priority, the device shall try to obtain data from the receivers. A client shall use GetRecordingJobState to determine if data transfer is really taking place. The only valid values for Mode shall be "Idle" and "Active".

Priority: This shall be a non-negative number. If there are multiple recording jobs that store data to the same track, the device will only store the data for the recording job with the highest priority. The priority is specified per recording job, but the device shall determine the priority of each track individually. If there are two recording jobs with the same priority, the device shall record the data corresponding to the recording job that was activated last.

The value 0 indicates the lowest priority. Higher values shall indicate a higher priority.

SourceToken: This field shall be a reference to the source of the data. The type of the source is determined by the attribute Type in the SourceToken structure. The value is given in the form of an URI[1]. This document defines the following two values for the attribute Type. If Type is http://www.onvif.org/ver10/schema/Receiver, the token is a ReceiverReference, see 10. In this case the device shall receive the data over the network. If Type is http://www.onvif.org/ver10/schema/Profile, the token identifies a media profile, see IEC 62676-11-31, instructing the device to obtain data from a profile that exists on the local device.

1 The URI is only used to create a unique identifier and is not useful for retrieving any document.

A device that includes support for the media service as defined in IEC 62676-11-31 shall support a media profile token and a device that includes support for the receiver service as defined in this document shall support a Receiver token.

If the SourceToken is omitted, AutoCreateRecevier shall be true.

AutoCreateReceiver: If this field is TRUE, and if the SourceToken is omitted, the device shall create a receiver object (through the receiver service) and assign the ReceiverReference to the SourceToken field. When retrieving the RecordingJobConfiguration from the device, the AutoCreateReceiver field shall never be present.

SourceTag: If the received RTSP stream contains multiple tracks of the same type, the SourceTag differentiates between those Tracks.

Destination: The destination is the track token of the track to which the device shall store the received data. All tracks shall belong to the recording identified by the RecordingToken.

The TrackInformation field for a Track holds a single Source. In the case where multiple RecordingJobs with differing sources are recording to the same track, it is undefined which of them is reported in the corresponding TrackInformation of the RecordingSearch API.

5.4 CreateRecording

CreateRecording shall create a new recording. The new recording shall be created with a track for each supported TrackType (see 5.21).

This method is optional. It shall be available if the Recording/DynamicRecordings capability is TRUE.

REQUEST:

RecordingConfiguration [tt:RecordingConfiguration]
Contains the initial configuration for the recording.

RESPONSE:

RecordingToken [tt:RecordingReference]
The reference to the created recording.

FAULTS:

env:Receiver – ter:Action – ter:MaxRecordings
The device cannot create a new recording because it already has the maximum number of recordings that it supports.

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
The RecordConfiguration is invalid.

env:Receiver – ter:ActionNotSupported – ter:NotImplemented
This optional method is not implemented.

ACCESS CLASS:

ACTUATE

When successfully completed, CreateRecording shall have created three tracks with the configurations shown in Table 2.

Table 2 – Track configuration

TrackToken	TrackType
VIDEO001	Video
AUDIO001	Audio
META001	Metadata

The RecordingConfiguration shall have the MaximumRetentionTime set to 0 (unlimited) and all TrackConfigurations shall have the Description set as an empty string.

5.5 DeleteRecording

DeleteRecording shall delete a recording object. Whenever a recording is deleted, the device shall delete all the tracks that are part of the recording, and it shall delete all the Recording Jobs that record into the recording. For each deleted recording job, the device shall also delete all the receiver objects associated with the recording job that are automatically created using the AutoCreateReceiver field of the recording job configuration structure and are not used in any other recording job.

This method is optional. It shall be available if the Recording/DynamicRecordings capability is TRUE.

REQUEST:

RecordingToken [tt:RecordingReference]
Identifies the recording that shall be deleted.

RESPONSE:

This is an empty message.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
RecordingToken does not reference an existing recording

env:Receiver – ter:ActionNotSupported – ter:NotImplemented
The device cannot delete recordings.

env:Receiver – ter:Action – ter:CannotDelete
This specific recording cannot be deleted.

ACCESS CLASS:

ACTUATE

5.6 GetRecordings

GetRecordings shall return a description of all the recordings in the device. This description shall include a list of all the tracks for each recording.

REQUEST:

This is an empty message.

RESPONSE:

RecordingItem – optional, unbounded [tt:GetRecordingsResponseItem]
Identifies a recording and its current configuration

FAULTS:

None

ACCESS CLASS:

READ_MEDIA

5.7 SetRecordingConfiguration

SetRecordingConfiguration shall change the configuration of a recording.

REQUEST:

RecordingToken [RecordingToken]
Identifies the recording that shall be changed.

RecordingConfiguration [RecordingConfiguration]
The new configuration for the recording.

RESPONSE:

This is an empty message.

FAULTS:

env:Sender – ter:InvalidArgVal – ter: BadConfiguration
The configuration is invalid.

env:Sender – ter:InvalidArgVal – ter:NoRecording
The RecordingToken does not reference an existing recording.

ACCESS CLASS:

ACTUATE

5.8 GetRecordingConfiguration

GetRecordingConfiguration shall retrieve the recording configuration for a recording.

REQUEST:

RecordingToken [tt:RecordingReference]
Identifies the recording for which the configuration shall be retrieved.

RESPONSE:

RecordingConfiguration [tt:RecordingConfiguration]
The current configuration for the requested recording.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
RecordingToken does not reference an existing recording.

ACCESS CLASS:

READ_MEDIA

5.9 CreateTrack

This method shall create a new track within a recording if the method GetRecordingOptions signals spare tracks for the recording. For a track to be created, the SpareXXX (where XXX is the track type) needs to be set.

This method is optional. It shall be available if the Recording/DynamicTracks capability is TRUE.

REQUEST:

RecordingToken [tt:RecordingReference]
Identifies the recording to which a track shall be added.

TrackConfiguration [tt:TrackConfiguration]
The configuration for the new track.

RESPONSE:

TrackToken [tt:TrackReference]
Identifies the newly created track. A device shall ensure that TrackToken is unique within the recording to which the new track belongs.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
RecordingToken does not reference an existing recording.

env:Receiver – ter:Action – ter:MaxTracks
The new track cannot be created because the maximum number of tracks that the device supports for this recording has been reached.

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
TrackConfiguration is invalid.

env:Receiver – ter:ActionNotSupported – ter:NotImplemented
This optional method is not implemented.

ACCESS CLASS:

ACTUATE

A TrackToken in itself does not uniquely identify a specific track. Tracks within different recordings may have the same TrackToken.

5.10 DeleteTrack

DeleteTrack shall remove a track from a recording. All the data in the track shall be deleted.

This method is optional. It shall be available if the Recording/DynamicTracks capability is TRUE.

REQUEST:

RecordingToken [tt:RecordingReference]
Identifies the recording from which to delete the track.

TrackToken [tt:TrackReference]
Identifies the track to delete.

RESPONSE:

This is an empty message.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
RecordingToken does not reference an existing recording.

env:Sender – ter:InvalidArgVal – ter:NoTrack
TrackToken does not reference an existing track of the recording.

env:Receiver – ter:Action – ter:CannotDelete
This specific track cannot be deleted.

env:Receiver – ter:ActionNotSupported – ter:NotImplemented
This optional method is not implemented.

ACCESS CLASS:

ACTUATE

5.11 GetTrackConfiguration

GetTrackConfiguration shall retrieve the configuration for a specific track.

REQUEST:

RecordingToken [tt:RecordingReference]
Identifies the recording.

TrackToken [tt:TrackReference]
Identifies the track within the recording from which to get the track configuration

RESPONSE:

TrackConfiguration [tt:TrackConfiguration]
The current configuration for the track.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
RecordingToken does not reference an existing recording.

env:Sender – ter:InvalidArgVal – ter:NoTrack
TrackToken does not reference an existing track of the recording.

ACCESS CLASS:

READ_MEDIA

5.12 SetTrackConfiguration

SetTrackConfiguration shall change the configuration of a track. TrackType shall be ignored by the device as it can't be changed. The TrackConfiguration is the new configuration for the track.

REQUEST:

RecordingToken [tt:RecordingReference]
Identifies the recording.

TrackToken [tt:TrackReference]
Identifies the recording within the recording from which to set the track configuration.

TrackConfiguration [tt:TrackConfiguration]
The new configurartion for the track.

RESPONSE:

This is an empty message.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
RecordingToken does not reference an existing recording.

env:Sender – ter:InvalidArgVal – ter:NoTrack
TrackToken does not reference an existing track of the recording.

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
The contents of the configuration object are invalid.

ACCESS CLASS:

ACTUATE

5.13 CreateRecordingJob

CreateRecordingJob shall create a new recording job. A device shall support adding a RecordingJob to a recording for which it signals Spare jobs via GetRecordingOptions.

REQUEST:

JobConfiguration [tt:RecordingJobConfiguration]
The configuration of the new recording job.

RESPONSE:

JobToken [tt:RecordingJobReference]
Identifies the created recording job.

JobConfiguration [tt:RecordingJobConfiguration]
The configuration as it used by the device. This may be different from the JobConfiguration passed to CreateRecordingJob.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
The RecordingToken given in the JobConfiguration does not reference an exsiting recording.

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
The contents of JobConfiguration are invalid.

env:Receiver – ter:Action – ter:MaxRecordingJobs
The maximum number of recording jobs that the device can handle has been reached.

env:Receiver – ter:Action – ter:MaxReceivers
If the AutoCreateReceivers flag is TRUE, this error can be returned if the receiver service cannot create a new receiver.

ACCESS CLASS:

ACTUATE

The JobConfiguration returned from CreateRecordingJob shall be identical to the JobConfiguration passed into CreateRecordingJob, except for the ReceiverToken and the AutoCreateReceiver. In the returned structure, the ReceiverToken shall be present and valid and the AutoCreateReceiver field shall be omitted.

5.14 DeleteRecordingJob

DeleteRecordingJob removes a recording job. It shall also implicitly delete all the receiver objects associated with the recording job that are automatically created using the AutoCreateReceiver field of the recording job configuration structure and are not used in any other recording job.

REQUEST:

JobToken [tt:RecordingJobReference]
Identifies the recording job that shall be deleted.

RESPONSE:

This is an empty message.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:NoRecordingJob
JobToken does not reference an existing job.

ACCESS CLASS:

ACTUATE

5.15 GetRecordingJobs

GetRecordingJobs shall return a list of all the recording jobs in the device.

REQUEST:

This is an empty message.

RESPONSE:

JobItem– optional, unbounded [tt:GetRecordingJobsResponseItem]
Identifies a job in the device and holds its current configuration.

FAULTS:

None

ACCESS CLASS:

READ_MEDIA

5.16 SetRecordingJobConfiguration

SetRecordingJobConfiguration shall change the configuration for a recording job. A device shall reject a request that tries to modify RecordingToken.

The JobConfiguration returned from SetRecordingJobConfiguration by a device shall be identical to the JobConfiguration passed into SetRecordingJobConfiguration, except for the ReceiverToken and the AutoCreateReceiver. In the returned structure, the ReceiverToken shall be present and valid and the AutoCreateReceiver field shall be omitted.

REQUEST:

JobToken [tt:RecordingJobReference]
Identifies the recording job to update.

JobConfiguration [tt:RecordingJobConfiguration]
The configuration to apply to the recording job.

RESPONSE:

JobConfiguration [tt:RecordingJobConfiguration]
The JobConfiguration structure shall be the configuration as it is used by the device. This may be different from the JobConfiguration passed to SetRecordingJobConfiguration.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:NoRecordingJob
JobToken does not reference an existing job.

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
The contents of JobConfiguration are invalid.

env:Receiver – ter:Action – ter:MaxReceivers
If the AutoCreateReceivers flag is TRUE, this error can be returned if the receiver service cannot create a new receiver.

ACCESS CLASS:

ACTUATE

SetRecordingJobConfiguration shall implicitly delete any receiver objects that were created automatically if they are no longer used as a result of changing the recording job configuration.

5.17 GetRecordingJobConfiguration

GetRecordingJobConfiguration shall return the current configuration for a recording job.

REQUEST:

JobToken [tt:RecordingJobReference]
Identifies the recording job for which to retrieve the configuration.

RESPONSE:

JobConfiguration [tt:RecordingJobConfiguration}
The current configuration of the recording job.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:NoRecordingJob
JobToken does not reference an existing job.

ACCESS CLASS:

READ_MEDIA

5.18 SetRecordingJobMode

SetRecordingJobMode shall change the mode of the recording job. Using this method shall be equivalent to retrieving the recording job configuration, and writing it back with a different mode.

Note that the state of a recording job will only become active if the recording job has the highest priority of all active jobs of a recording.

REQUEST:

JobToken [tt:RecordingJobReference]
Identifies the recording job for which to change the recording mode.

Mode [tt:RecordingJobMode]
The new mode for the recording job.

RESPONSE:

This is an empty message.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:NoRecordingJob
JobToken does not reference an existing job.

env:Sender – ter:InvalidArgVal – ter:BadMode
The Mode is invalid.

ACCESS CLASS:

ACTUATE

5.19 GetRecordingJobState

GetRecordingJobState returns the state of a recording job. It includes an aggregated state, and state for each track of the recording job. The RecordingJogState may change due to

- calls that effect the RecordingJobMode, e.g. SetRecordingJobMode,
- internal recording engine state changes,
- changes in the recorded local media profile, or
- changes to the RTSP connection defined by the associated Receiver.

REQUEST:

JobToken [tt:RecordingJobReference]
Identifies the recording job for which to get the state.

RESPONSE:

State [tt:RecordingJobReference]
The state of the recording job.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:NoRecordingJob
JobToken does not reference an existing job.

ACCESS CLASS:

READ_MEDIA

The UML representation of the RecordingJobStateInformation structure is shown in Figure 5.

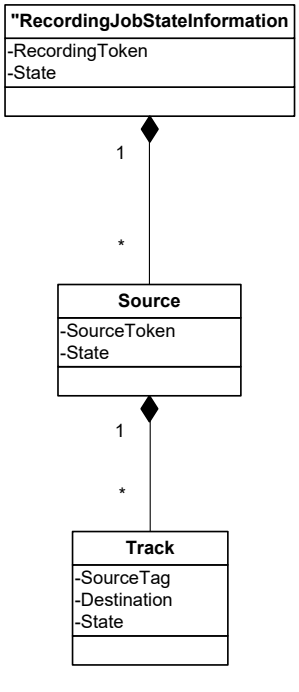

Figure 5 – RecordingJobStateInformation structure

RecordingToken shall be the identification of the recording that the recording job records to.

State (as part of RecordingJobStateInformation) shall hold the aggregated state over the whole RecordingJobInformation structure.

SourceToken shall identify the data source of the recording job.

State (as part of RecordingJobStateSource) shall hold the aggregated state over all substructures of RecordingJobStateSource.

SourceTag shall identify the track of the data source that provides the data.

Destination shall indicate the destination track

State (as part of RecordingJobTrackState) shall provide the job state of the track. The valid values of state shall be "Idle", "Active" and "Error". If state equals "Error", the Error field may be filled in with an implementation defined value.

Error, optional string describing the error state. The string should be in English. The following values are predefined:

"Incompatible Stream" – The stream cannot be recorded because the encoding does not match to previously recorded data.

A device shall apply the following rules to compute the aggregate state:

Idle	All state values in sub-nodes are "Idle"
PartiallyActive	The state of some sub-nodes are "Active" and some sub-nodes are "Idle"
Active	The state of all sub-nodes is "Active"
Error	At least one of the sub-nodes has the state "Error"

5.20 GetRecordingOptions

GetRecordingOptions returns information for a recording identified by the RecordingToken. The information includes the number of additional tracks as well as recording jobs that can be configured.

This method shall be supported if the Options support is signaled via the capabilities.

Note that this information is not static and is only guaranteed to be valid until the next modification of any recording jobs or tracks.

The track options shall be supported if the device signals support for dynamic tracks.

REQUEST:

RecordingToken [tt:RecordingReference]
Identifies the recording.

RESPONSE:

JobOptions [trc:JobOptions]
Contains two attributes:

Spare: Number of spare jobs that can be created for the recording. By setting none of the Spare attribute, the device signals that no job can be created.

CompatibleSources: A device that supports recording of a restricted set of media service profiles shall return the list of profiles that can be recorded on the given recording.

TrackOptions [trc:TrackOptions]
Contains four attributes:

SpareTotal: Total spare number of tracks that can be added to this recording.

SpareVideo: Number of spare video tracks for this recording

SpareAudio: Number of spare audio tracks for this recording

SpareMetadata: Number of spare metadata tracks for this recording

By setting none of the SpareXXX attributes, the device signals that no track can be added.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
RecordingToken does not reference an existing recording.

ACCESS CLASS:

READ_MEDIA

5.21 ExportRecordedData

ExportRecordedData exports the selected recordings to the given storage target.

A device that indicates a capability of SupportedExportFileFormats shall support this command. For the parameter FileFormat, it shall accept any format that it advertises via the SupportedExportFileFormats capability.

REQUEST:

StartPoint – optional [xs:dateTime]
Specifies start time for the exporting.

EndPoint – optional [xs:dateTime]
Specifies end time for the exporting.

SearchScope [tt:SearchScope]
Defines the selection criterion for the existing recordings.

FileFormat [xs:string]
Indicates which export file format to be used.

StorageDestination [tt:StorageReferencePath]
Indicates the target storage and relative directory path.

RESPONSE:

OperationToken [tt:ReferenceToken}
The asynchronous operation token for associating the received event with this invocation.

FileNames – optional, unbounded [xs:string]
List of exported file names. The device can also use AsyncronousOperationStatus event to monitor the progress of the ExportRecordedData operation.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:BadConfiguration:
The device cannot support the selected FileFormat for exported files.

ACCESS CLASS:

READ_MEDIA

A device should return the resulting list of file names in the response. In cases where the retrieval of the file names is a longer-lasting operation, these file names may be reported only via the AsyncronousOperationStatus.

The ExportRecordedDataResponse returns the unique operation token that is used by client to monitor the progress of this invocation. The progress of export recordings operation is obtained via an event (Core Spec, Monitoring/AsynchronousOperationStatus). A device can notify the progress of exported files via the optional FileProgressStatus (tt:ArrayOfFileProgress) element, containining an array of file name and completion percentage of file upload from the device's point of view. The value of completion percentage is normalized between 0 and 1, where 1 indicates 100 % completion of the file upload. The optional FileProgressStatus element is sent in the Data section of a message.

If a delete recording request is issued during an export of recordings and there are common recordings, the device shall delete the relevant recording after completing the export of these relevant recordings.

5.22 StopExportRecordedData

Stops the ExportRecordedData operation that is started before. The response message lists the status of the exported files.

A device that indicates a capability of SupportedExportFileFormats shall support this command.

REQUEST:

OperationToken [tt:ReferenceToken]
Identifies the ExportRecordedData operation to stop.

RESPONSE:

Progress [xs:float]
Completion percentage of total file upload from the device's point of view. The value of the completion percentage is normalized between 0,0 and 1,0, where 1,0 indicates 100 % completion of the total file upload.

FileProgressStatus [tt:ArrayOfFileProgress]
An array of file names and an individual progress for each file. The value of completion percentage is normalized between 0,0 and 1,0, where 1,0 indicates 100 % completion of the file upload.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
The requested operation does not exist.

ACCESS CLASS:

READ_MEDIA

5.23 GetExportRecordedDataState

GetExportRecordedDataState returns the status of export operations. This interface allows the client to poll the status information from the device.

A device that indicates a capability of SupportedExportFileFormats shall support this command.

REQUEST:

OperationToken [tt:ReferenceToken]
Identifies the ExportRecordedData operation from which to get status.

RESPONSE:

Progress [xs:float]
Completion percentage of total file upload from the device's point of view. The value of completion percentage is normalized between 0,0 and 1,0 where, 1,0 indicates 100 % completion of total file upload.

FileProgressStatus [tt:ArrayOfFileProgress]
An array of file names and an individual progress for each file. The value of completion percentage is normalized between 0,0 and 1,0, where 1,0 indicates 100 % completion of the file upload.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
The requested operation does not exist.

ACCESS CLASS:

READ_MEDIA

5.24 GetServiceCapabilities

The capabilities reflect optional functions and functionality of a service. The information is static and does not change during device operation. The following capabilities are available:

DynamicRecordings Indication if the device supports dynamic creation and deletion of recordings.

DynamicTracks Indication if the device supports dynamic creation and deletion of tracks.

Encoding Indication which encodings are supported for recording. The list may contain one or more enumeration values of tt:VideoEncoding and tt:AudioEncoding. For encodings that are neither defined in tt:VideoEncoding nor tt:AudioEncoding the device shall use the IANA definitions see [IANA Media Types]. Note that a device without audio support shall not return audio encodings.

MaxRate Maximum supported bit rate for all tracks of a recording in kBit/s.

MaxTotalRate Maximum supported bit rate for all recordings in kBit/s.

MaxRecordings Maximum number of recordings supported.

MaxRecordingJobs Maximum total number of supported recording jobs by the device.

Options Indication if the device supports the GetRecordingOptions command.

MetadataRecording Indication if the device supports recording metadata.

SupportedExportFileFormats Indication that the device supports the ExportRecordedData command for the listed export file formats. A device shall ony return this capability if it contains at least one export file format value. The value of 'ONVIF' refers to the Export File Format Specification as defined in this document.

REQUEST:

This is an empty message.

RESPONSE:

Capabilities [trc:Capabilities]
List of capabilities as defined above.

FAULTS:

None

ACCESS CLASS:

PRE_AUTH

5.25 Events

5.25.1 General

Some of these events are similar to the automatically generated events that can be searched for by the FindEvents method in the search service (see Clause 6).

5.25.2 Recording job state changes

If the state field of the RecordingJobStateInformation structure changes, a device shall provide the following event:

```
Topic: tns1:RecordingConfig/JobState
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingJobToken"
Type="tt:RecordingJobReference"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="xs:String"/>
    <tt:ElementItemDescription Name="Information"
Type="tt:RecordingJobStateInformation"/>
  </tt:Data>
</tt:MessageDescription>
```

5.25.3 Configuration changes

If the configuration of a recording is changed, a device shall provide the following event:

```
Topic: tns1:RecordingConfig/RecordingConfiguration
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Configuration" Type="tt:RecordingConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

If the configuration of a track is changed, a device shall provide the following event:

```
Topic: tns1:RecordingConfig/TrackConfiguration
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
    <tt:SimpleItemDescription Name="TrackToken" Type="tt:TrackReference"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Configuration" Type="tt:TrackConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

If the configuration of a recording job is changed, a device shall provide the following event:

```
Topic: tns1:RecordingConfig/RecordingJobConfiguration
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingJobToken"
Type="tt:RecordingJobReference"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Configuration"
Type="tt:RecordingJobConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

5.25.4 Data deletion

Whenever data is deleted, a device shall provide the following event:

```
Topic: tns1:RecordingConfig/DeleteTrackData
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
    <tt:SimpleItemDescription Name="TrackToken" Type="tt:TrackReference"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="StartTime" Type="xsDateTime"/>
    <tt:SimpleItemDescription Name="EndTime" Type="xsDateTime"/>
  </tt:Data>
</tt:MessageDescription>
```

5.25.5 Recording and track creation and deletion

Whenever a recording is created, a device shall provide the following event:

```
Topic: tns1:RecordingConfig/CreateRecording
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
  </tt:Source>
  <tt:Data>
  </tt:Data>
</tt:MessageDescription>
```

Whenever a recording is deleted, a device shall provide the following event:

```
Topic: tns1:RecordingConfig/DeleteRecording
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
  </tt:Source>
  <tt:Data>
  </tt:Data>
</tt:MessageDescription>
```

Whenever a track is created, a device shall provide the following event:

```
Topic: tns1:RecordingConfig/CreateTrack
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
    <tt:SimpleItemDescription Name="TrackToken" Type="tt:TrackReference"/>
  </tt:Source>
  <tt:Data>
  </tt:Data>
</tt:MessageDescription>
```

Whenever a track is deleted, a device shall provide the following event:

```
Topic: tns1:RecordingConfig/DeleteTrack
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
    <tt:SimpleItemDescription Name="TrackToken" Type="tt:TrackReference"/>
  </tt:Source>
  <tt:Data>
  </tt:Data>
</tt:MessageDescription>
```

5.26 Examples

5.26.1 Example 1: setup recording of a single camera

There are two steps involved. The first step is to configure the NVS

```
; Create recording (this implicitly creates an A, V and M track)
RecordToken = CreateRecording(RecordConfiguration)

; The tracktokens are predefined. We don't have to find them on the
device
TrackToken1 = "VIDEO001"
TrackToken2 = "AUDIO001"
TrackToken3 = "META001"

; Create a recording job, assume that we set mode to idle, auto create
receiver
JobToken, ActualJobConfig = CreateRecordingJob(JobConfiguration)

; Configure the receiver
ConfigureReceiver(ActualJobConfiguration.ReceiverToken,
ReceiverConfiguration)
```

This completes the configuration step.

Finally, to really start recording, some entity calls

```
; Activate the recording job
SetRecordingJobMode(JobToken, Active)
```

to make the job active. This will cause the NVS to set up an RTSP connection with the device.

Therefore, to start and stop recording, all that is needed is to call SetRecordingJobMode on pre-configured recording jobs. And since the embedded configuration objects are persistent, the configuration cycle only needs to be done once.

5.26.2 Example 2: Record multiple streams from one camera to a single recording

This example is very similar to example 1 (see 5.26.1). The job configuration will hold references to two receiver objects. Each receiver object is configured to receive from the same device, but from a different stream.

```
      ; Create recording (this implicitly creates an A, V and M track)
      RecordToken = CreateRecording(RecordConfiguration)

      ; The tracktokens are predefined. We don't have to find them on the
device
      TrackToken1 = "VIDEO001"
      TrackToken2 = "AUDIO001"
      TrackToken3 = "META001"

      ; Create three additional tracks
      TrackToken4 = CreateTrack(RecordToken, AudioConfig)
      TrackToken5 = CreateTrack(RecordToken, VideoConfig)
      TrackToken6 = CreateTrack(RecordToken, MetadataConfig)


      ; Create a recording job, assume that we set mode to idle,auto create
two receivers
      JobToken, ActualJobConfiguration = CreateRecordingJob(JobConfiguration)


      ; Configure the receivers
      ConfigureReceiver(ActualJobConfiguration.ReceiverToken[1],
                                                  Receiver1Configuration)
      ConfigureReceiver(ActualJobConfiguration.ReceiverToken[2],
                                                  Receiver2Configuration)
```

To really start recording, some entity calls

```
      ; Activate the recording job
      SetRecordingJobMode(JobToken, Active)
```

6 Search service

6.1 General

The search service provides a number of operations for finding data of interest within a set of recordings. The most common way of doing this would be to search for events that are either included in the metadata track of a recording, or are otherwise associated with a recording in the device (see 6.2.2).

GetRecordingSummary returns a summary for all recording, and can be used to provide the scale of a timeline.

GetRecordingInformation returns information about a single recording, such as start time and current status.

GetMediaAttributes returns the media attributes of a recording at a specific point in time.

The actual search is done by coupled find and result operations. Each find operation initiates a search session. The client can then acquire the results from the search session in increments, or all at once, depending on implementation and the scale of the search. There are four pairs of search operations for recordings, recording events, PTZ positions and metadata.

EndSearch ends a search session, halting search and returning any blocking result operations.

6.2 Concepts

6.2.1 Search direction

Search is performed from a start point on the timeline towards an end point. If the end point is prior to the start point, the search will be performed backwards. This can be useful if only the newest matching event is of interest, or if it is otherwise convenient to get the results in newest to oldest order.

If no end point is specified, the search will always be performed forward in time from the start point.

6.2.2 Recording event

Describes a discrete event related to the recording. It is represented as a notification message, but this does not necessarily mean it has been recorded as a notification. Recording events can be notifications included in a recorded metadata track, or they can be created by the recording device as a result of an internal event or mechanism, or they can be inserted by a client using a WebService request or a metadata stream. However, the recording event has been created and associated with a particular recording, this specification makes no implications on how it is stored internally on a device, only how it should be represented in the interface.

However created, recording events are always treated as notifications in regards to search filters and results returned. Each recording event has a notification topic as defined in 10.7 of IEC 60839-11-31:2016. Predefined recording events are described in 6.17.

To communicate the original state of property events, virtual start state events can be returned in a search result containing the value of one or more properties at the start point of the search interval. Such start state events are virtual events in the sense that they are created on the fly by the server, rather than being collected from recorded data. If the client indicates that such events are desired by setting the appropriate flag, virtual events matching the topics defined in the search filter shall be returned for any recording in the search scope.

6.2.3 Search session

A search session is started asynchronously by a find operation and is identified by a search token unique for that session. Results are returned in increments using GetResult operations referring to the session created by the Find operation. The search can be terminated in three ways:

- KeepAlive time expires – If no request from a client has been made that refers to a particular session within the specified time interval, it will terminate.
- A GetResult method returns the last data for the search session by setting the search state in its result to "Completed".
- EndSearch – The client explicitly ends a session.

Ending a session will cancel an ongoing search, immediately return and make further requests to the same session result in an error message. A device shall not reuse search token immediately as it would confuse clients unaware that a session had ended.

6.2.4 Search scope

6.2.4.1 General

The scope contains a number of optional elements, together limiting the set of data to look into when performing searches.

6.2.4.2 Included data

Optionally, the client can define sources and recordings to search in by specifying lists of tokens for each type. If several types are given, the union of the specified tokens shall be used. If there are no sources or recordings tokens specified, all recordings shall be included. The scope is further refined by the Recording Information Filter. However, if recordings are specified, the filters will only be applied to that subset of recordings.

6.2.4.3 Recording information filter

Rather than specifying a list of recording tokens, the recordings can be filtered by an XPath filter operating on the RecordingInformation structure. This allows the client to filter on all elements present in the RecordingInformation structure, using comparisons according to the XPath dialect defined in 6.18. If a recording information filter is supplied, only recordings matching the filter shall be part of the scope.

Example of a filter that includes only recordings containing audio in the search scope:

```
boolean(//Track[TrackType = "Audio"])
```

6.2.5 Search filters

Search filters are specific for the type of search operation. See FindEvents, FindPTZPosition, and FindMetadata. They all act on the recordings defined by the scope.

6.2.6 Time information

A device shall support time values in request parameters that are given in UTC with the 'Z' indicator and respond to all time values as UTC, including the 'Z' indicator.

6.3 Data structures

6.3.1 RecordingInformation structure

RecordingInformation contains information about a recording, the tracks it consists of and the source.

- RecordingToken – a unique identifier of the recording.
- EarliestRecording – the date and time of the oldest data in the recording.
- LatestRecording – the date and time of the newest data in the recording.
- Content – informative description of content. If the content is unknown this field is empty
- RecordingStatus – current status of recording, can be any of: Initiated, Recording, Stopped, Removing, Removed, Unknown.
- RecordingSourceInformation – a structure containing information about the source of the recording.
- TrackInformation – a list of track information structures.

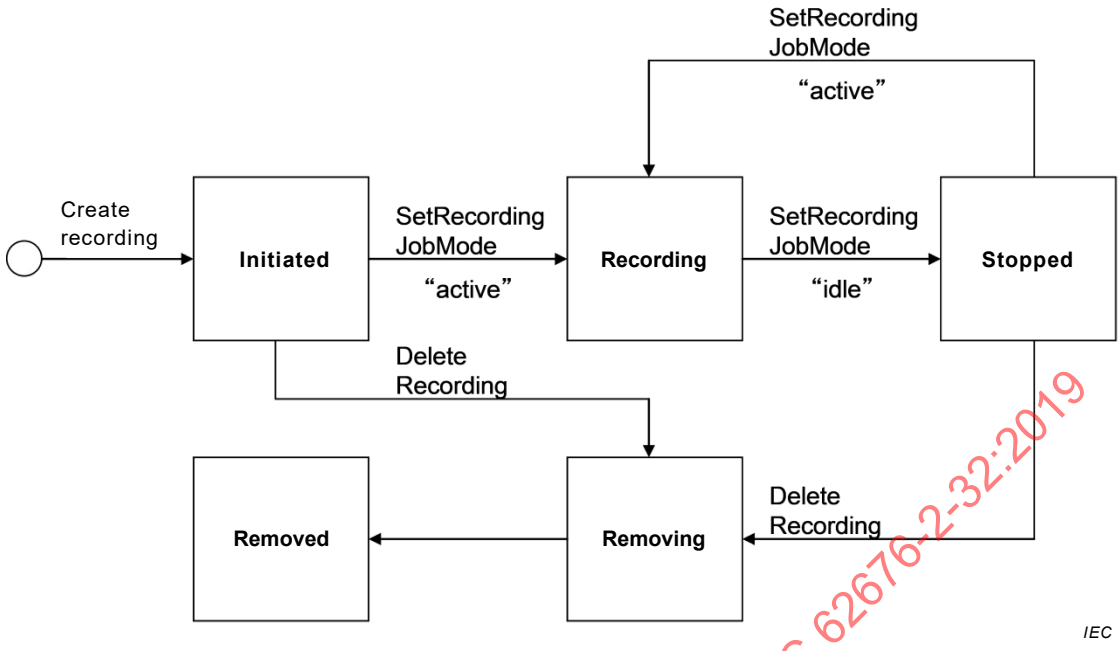

Figure 6 – Recording state chart

Figure 6 shows the state changes for the RecordingStatus. The following adds additional explanation:

- Initiated – the new recording is created and the specified recording has not yet started to record.
- Recording – the recording job is executing the specified recording.
- Stopped – the recording job is pausing the specified recording. This case requires that the job has started recording at least once before.
- Removing – the specified recording is in the process of being deleted.
- Removed – the specified recording has been removed.
- Unknown – this case should never happen.

6.3.2 RecordingSourceInformation structure

Contains information about the source of a recording.

- SourceId – an identifier for the source chosen by the client that creates the recording. This identifier is opaque to the device. Clients may use any type of URI for this field. If this identifier is not present, this field is empty.
- Name – informative name of the source. If the name is unknown, this field is empty.
- Location – informative description of the location of the source. If the description is not available, this string is empty.
- Description – informative description of the source. If this description is not present, the description is empty.
- Address – informative URI of the source.

6.3.3 TrackInformation structure

Contains information about a single track in a recording. Note that a track may represent a single contiguous time span or consist of multiple slices as shown in 5.1. If there is no recorded data for a track, TrackInformation shall not be provided.

- TrackToken – an identifier of the track. TrackToken is unique between all TrackTokens used within a recording.
- TrackType – identifies the type of track (video, audio or metadata).
- Description – informative description of the track. If this information is not present, this field is empty.
- DataFrom – The start date and time of the oldest recorded data in the track.
- DataTo – The stop date and time of the newest recorded data in the track.

6.3.4 SearchState Enumeration

The search state can be one of the following

- Searching – The database search is in progress and there may be results available that can be fetched via the method GetEventSearchResults.
- Completed – Search has been completed and all results have been delivered via GetEventSearchResults.

The usage of the additionally defined state "Queued" is deprecated.

6.3.5 MediaAttributes structure

MediaAttributes contains information about the media tracks of a particular recording for a particular time frame. The time frame can be a single point in time, in which case the From and Until elements are identical.

- RecordingToken – a reference to the recording that this structure concerns.
- From – a point in time from when the attributes are valid for the recording.
- Until – a point in time until when the specified attributes are valid for the recording.
- VideoAttributes – a set of video attributes, describing the data of a recorded video track.
- AudioAttributes – a set of audio attributes, describing the data of a recorded audio track.
- MetadataAttributes – a set of attributes, describing the possible metadata content of a recorded metadata track.

6.3.6 FindEventResult structure

- RecordingToken – identifying the recording containing the found event.
- TrackToken – identifying the track containing the found event.
- Time – the date and time of the event found.
- Event – the event message found.
- StartStateEvent – if true, indicates the event represents the start state of one or more properties in the recording.

6.3.7 FindPTZPositionResult structure

- RecordingToken – identifying the recording containing the matching position.
- TrackToken – identifying the track containing the matching position.
- Time – the date and time of the matching position.
- Position – the matching PTZ vector.

6.3.8 PTZPositionFilter structure

Contains the necessary elements to define what PTZ positions to search for. The PTZ vectors shall be in the same coordinate space as the PTZ coordinates stored in the recording.

- MinPosition – the lower boundary of the PTZ volume to look for.

- MaxPosition – the upper boundary of the PTZ volume to look for.
- EnterOrExit – If true, report the positions when entering or exiting the specified PTZ volume. Otherwise, report all recorded positions within the specified PTZ range.

6.3.9 MetadataFilter structure

Contains an XPath expression to be applied to the MetadataStream structure.

Example of an expression searching for objects overlapping the lower right quadrant of the scene:

```
boolean(//Object/Appearance/Shape/BoundingBox[@right > "0.5"])
and boolean(//Object/Appearance/Shape/BoundingBox[@bottom > "0.5"])
```

6.3.10 FindMetadataResult structure

- RecordingToken – identifying the recording containing the matching metadata.
- TrackToken – identifying the track containing the matching metadata.
- Time – the date and time of the matching metadata.

6.4 GetRecordingSummary

GetRecordingSummary is used to get a summary description of all recorded data. This operation is mandatory to support a device implementing the recording search service. If a device returns NumberRecordings as zero, both DataFrom and DataUntil can be safely ignored.

REQUEST:

This is an empty message.

RESPONSE:

Summary [tt:RecordingSummary]
A structure containing: DataFrom specifying the first time when there is recorded data on the device; DataUntil specifying the last point in time where there is data recorded on the device; the total number of recordings on the device.

FAULTS:

None

ACCESS CLASS:

READ_MEDIA

6.5 GetRecordingInformation

Returns information about a single Recording specified by a RecordingToken. This operation is mandatory to support for a device implementing the recording search service.

REQUEST:

RecordingToken [tt:ReferenceToken]
Identifies the recording.

RESPONSE:

RecordingInformation [tt:RecordingInformation]
Information about the recording.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:InvalidToken
The recording token is not valid.

ACCESS CLASS:

READ_MEDIA

6.6 GetMediaAttributes

Returns a set of media attributes for all tracks of the specified recordings at a specified point in time. Clients using this operation shall be able to use it as a non-blocking operation. A device shall set equal values for the start time and the end time of the MediaAttributes structure if calculating this range would cause this operation to block. See MediaAttributes structure for more information. This operation is mandatory for supporting a device implementing the recording search service.

Devices indicating CanContainPTZ shall report the PTZ spaces in use at the specified point in time (including generic spaces). For optimal interoperability, device implementations should use generic spaces[2]. The generic spaces are the following, see 8.9.2 of IEEE 62676-2-31:2019:

http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace

http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace

REQUEST:

RecordingTokens – optional, unbounded [tt:ReferenceToken]
A list of references to the recordings to query. If no recording token is provided, all recordings should be queried.

Time [xs:dateTime]
The point in time from where the information is requested.

RESPONSE:

MediaAttributes – optional, unbounded [tt:MediaAttributes]
Contains a MediaAttributes structure for the RecordingToken specified in the request. Note that each RecordingToken can result in zero or one MediaAttributes.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:InvalidToken
The recording token is not valid.

ACCESS CLASS:

READ_MEDIA

[2] A PTZ space is identified using an URI. The URI is only used to create a unique identifier and is not useful for retrieving any document.

6.7 FindRecordings

FindRecordings starts a search session, looking for recordings that matches the scope (see 6.2.4) defined in the request. Results from the search are acquired using the GetRecordingSearchResults request, specifying the search token returned from this request.

The device shall continue searching until one of the following occurs:

- the total number of matches has been found, defined by the MaxMatches parameter;
- the session has been ended by a client EndSearch request;
- the session has been ended because KeepAliveTime since the last request related to this session has expired.

The order of the results is undefined, to allow the device to return results in any order they are found. This operation is mandatory for supporting a device implementing the recording search service.

For KeepAliveTime, a device shall support at least values up to 10 s. A device may adapt larger values.

REQUEST:

Scope [tt:SearchScope]
Defines the dataset to consider for this search.

MaxMatches – optional [xs:int]
The search ends after MaxMatches.

KeppAliveTimeout [xs:duration]
The session timeout after each request concerning this session.

RESPONSE:

SearchToken [tt:Jobtoken]
Identifies the search session created by this request.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:InvalidToken
The recording token is not valid.

env:Sender – ter:InvalidArgVal – ter:InvalidSource
The recording source is not valid.

env:Receiver – ter:Action – ter:ResourceProblem
Device is unable to create a new search session.

ACCESS CLASS:

READ_MEDIA

6.8 GetRecordingSearchResults

GetRecordingSearchResults acquires the results from a recording search session previously initiated by a FindRecordings operation. The response shall not include results already returned in previous requests for the same session. If MaxResults is specified, the response shall not contain more than MaxResults results. The number of results relates to the number of recordings. For viewing individual recorded data for a signal track, use the FindEvents method.

GetRecordingSearchResults shall block until:

- MaxResults results are available for the response if MaxResults is specified;
- MinResults results are available for the response if MinResults is specified;
- WaitTime has expired;
- Search is completed or stopped.

This operation is mandatory to support for a device implementing the recording search service. If any of the specified parameters MinResults and WaitTime exceed the supported range, a device shall adapt them instead of responding an error.

REQUEST:

SearchToken [tt:JobToken]
Specifies the search session.

MinResults – optional [xs:int]
Specifies the minimum number of results that should be returned. If the total number of results is lower than MinResults in a completed search, all results should be returned.

MaxResults – optional [xs:int]
Specifies the maximum number of results to return.

WaitTime – optional [xs:duration]
Defines the maximum time to block, waiting for results.

RESPONE:

ResultList [tt:FindRecordingResultList]
A structure containing the current SearchState and a list of RecordingInformation structures.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:InvalidToken
The search token is invalid.

ACCESS CLASS:

READ_MEDIA

6.9 FindEvents

FindEvents starts a search session, looking for events in the scope (see 6.2.4) that match the search filter defined in the request. Events are recording events (see 6.2.2) and other events that are available in the track. Results from the search are acquired using the GetEventSearchResults request, specifying the search token returned from this request.

The device shall continue searching until one of the following occurs:

- the entire time range from StartPoint to EndPoint has been searched through;
- the total number of matches has been found, defined by the MaxMatches parameter;
- the session has been ended by a client EndSearch request;
- the session has been ended because KeepAliveTime since the last request related to this session has expired.

Results shall be ordered by time, ascending in case of forward search, or descending in case of backward search. This operation is mandatory for supporting a device that implements the

recording search service. Although the values of property events refer to the forward direction, they shall be reported identically in reverse search mode.

For KeepAliveTime, a device shall support at least values up to 10 s. A device may adapt larger values.

REQUEST:

StartPoint [xs:dateTime]
The point of time where the search will start.

EndPoint – optional [xs:dateTime]
The point of time where the search will stop. This can be a time before the StartPoint, in which case the search is performed backwards in time. If EndPoint is omitted, search will go forward from the StartPoint.

Scope [tt:SearchScope]
Defines the dataset to consider for this search.

SearchFilter [tt:EventFilter]
Contains the topic and message filter needed to define what events to search for.

IncludeStartState [xs:boolean]
By setting the IncludeStartState to true, the client indicates that virtual events at the time of StartPoint should be returned to represent the state in the recording. In the case of a backward search, virtual events at the time of EndPoint and StartPoint should be returned. Support for virtual events is mandatory for recording events. Support for additional virtual events is signalled via the GeneralStartEvents capability.

MaxMatches – optional [xs:int]
The search ends after MaxMatches.

KeepAliveTime [xs:duration]
The session timeout after each request concerning this session.

RESPONSE:

SearchToken [tt:JobToken]
Identifies the search session created by this request.

FAULTS:

env:Receiver – ter:Action – ter:ResourceProblem
Device is unable to create a new search session.

env:Sender – ter:InvalidArgVal – ter:InvalidFilterFault
Provided search filter expression was not understood or supported by the device.

ACCESS CLASS:

READ_MEDIA

6.10 GetEventSearchResults

GetEventSearchResults acquires the results from a recording event search session previously initiated by a FindEvents operation. The response shall not include results already returned in previous requests for the same session. If MaxResults is specified, the response shall not contain more than MaxResults results.

GetEventSearchResults shall block until:

- MaxResults results are available for the response if MaxResults is specified;

- MinResults results are available for the response if MinResults is specified;
- WaitTime has expired;
- search is completed or stopped.

This operation is mandatory to support for a device implementing the recording search service. If any of the specified parameters MinResults and WaitTime exceed the supported range, a device shall adapt them instead of responding with an error.

REQUEST:

SearchToken [tt:JobToken]
Specifies the search session.

MinResults – optional [xs:int]
Specifies the minimum number of results that should be returned. If the total number of remaining event is lower than MinResults in a completed search, all remaining events should be returned.

MaxResults – optional [xs:int]
Specifies the maximum number of results to return.

WaitTime – optional [xs:duration]
Defines the maximum time to block, waiting for results.

RESPONSE:

ResultList [tt: FindEventResultList]
A structure containing:

The state of the search when the result is returned. Indicates if there can be more results, or if the search is completed.

A FindEventResult structure for each found event matching the search.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:InvalidToken
The search token is invalid.

ACCESS CLASS:

READ_MEDIA

6.11 FindPTZPosition

FindPTZPosition starts a search session, looking for PTZ positions in the scope (see 6.2.4) that matches the search filter defined in the request. Results from the search are acquired using the GetPTZPositionSearchResults request, specifying the search token returned from this request.

The device shall continue searching until one of the following occurs:

- the entire time range from StartPoint to EndPoint has been searched through;
- the total number of matches has been found, defined by the MaxMatches parameter;
- the session has been ended by a client EndSearch request;
- the session has been ended because KeepAliveTime since the last request related to this session has expired.

This operation is mandatory to support whenever CanContainPTZ is true for any metadata track in any recording on the device.

For KeepAliveTime, a device shall support at least values up to 10 s. A device may adapt larger values.

A device shall only match the search criteria against PTZ status updates available between the time interval given in the search, i.e. the device shall not locate the PTZ position at the start of the search interval.

REQUEST:

StartPoint [xs:dateTime]
The point of time where the search will start.

EndPoint – optional [xs:dateTime]
The point of time where the search will stop. This can be a time before the StartPoint, in which case the search is performed backwards in time. If EndPoint is omitted, search will go forward from the StartPoint.

Scope [tt:SearchScope]
Defines the dataset to consider for this search.

SearchFilter [tt: PTZPositionFilter]
Contains the search criteria needed to define the PTZ position to search for.

MaxMatches – optional [xs:int]
The search ends after MaxMatches.

KeepAliveTime [xs:duration]
The session timeout after each request concerning this session.

RESPONSE:

SearchToken [tt:JobToken]
Identifies the search session created by this request.

FAULTS:

env:Receiver – ter:Action – ter:ResourceProblem
Device is unable to create a new search session.

env:Receiver – ter:ActionNotSupported – ter:NotImplemented
This optional method is not implemented.

ACCESS CLASS:

READ_MEDIA

6.12 GetPTZPositionSearchResults

GetPTZPositionSearchResults acquires the results from a PTZ position search session previously initiated by a FindPTZPosition operation. The response shall not include results already returned in previous requests for the same session. If MaxResults is specified, the response shall not contain more than MaxResults results.

GetPTZPositionSearchResults shall block until:

- MaxResults results are available for the response if MaxResults is specified;
- MinResults results are available for the response if MinResults is specified;
- WaitTime has expired;
- search is completed or stopped.

Support for this operation is mandatory whenever CanContainPTZ is true for any metadata track in any recording on the device. If any of the specified parameters MinResults and WaitTime exceed the supported range, a device shall adapt them instead of responding with an error.

REQUEST:

SearchToken [tt:JobToken]
Specifies the search session.

MinResults – optional [xs:int]
Specifies the minimum number of results that should be returned. If the total number of remaining event is lower than MinResults in a completed search, all remaining events should be returned.

MaxResults – optional [xs:int]
Specifies the maximum number of results to return.

WaitTime – optional [xs:duration]
Defines the maximum time to block, waiting for results.

RESPONSE:

ResultList [tt:FindPTZPositionResultList]
A structure containing:

The state of the search when the result is returned. Indicates if there can be more results, or if the search is completed.

A FindPTZPositionResult structure for each found PTZ position matching the search.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:InvalidToken
The search token is invalid.

ACCESS CLASS:

READ_MEDIA

6.13 FindMetadata

FindMetadata starts a search session, looking for metadata in the scope (see 6.2.4) that matches the search filter defined in the request. Results from the search are acquired using the GetMetadataSearchResults request, specifying the search token returned from this request.

The device shall continue searching until one of the following occurs:

- the entire time range from StartPoint to EndPoint has been searched through;
- the total number of matches has been found, defined by the MaxMatches parameter;
- the session has been ended by a client EndSearch request;
- the session has been ended because KeepAliveTime since the last request related to this session has expired.

Support for this operation is mandatory if the MetaDataSearch capability is set to true in the SearchCapabilities structure return by the GetCapabilities command in the device service.

For the KeepAliveTime, a device shall support at least values up to 10 s. A device may adapt larger values.

REQUEST:

StartPoint [xs:dateTime]
The point of time where the search will start.

EndPoint – optional [xs:dateTime]
The point of time where the search will stop. This can be a time before the StartPoint, in which case the search is performed backwards in time. If EndPoint is omitted, search will go forward from the StartPoint.

Scope [tt:SearchScope]
Defines the dataset to consider for this search.

SearchFilter [tt: MetadataFilter]
Contains the search criteria needed to define the metadata to search for.

MaxMatches – optional [xs:int]
The search ends after MaxMatches.

KeepAliveTime [xs:duration]
The session timeout after each request concerning this session.

RESPONSE:

SearchToken [tt:JobToken]
Identifies the search session created by this request.

FAULTS:

env:Receiver – ter:Action – ter:ResourceProblem
Device is unable to create a new search session.

ACCESS CLASS:

READ_MEDIA

6.14 GetMetadataSearchResults

GetMetadataSearchResults acquires the results from a recording search session previously initiated by a FindMetadata operation. The response shall not include results already returned in previous requests for the same session. If MaxResults is specified, the response shall not contain more than MaxResults results.

GetMetadataSearchResults shall block until:

- MaxResults results are available for the response if MaxResults is specified;
- MinResults results are available for the response if MinResults is specified;
- WaitTime has expired;
- search is completed or stopped.

Support for this operation is mandatory if the MetaDataSearch capability is set to true in the SearchCapabilities structure returned by the GetCapabilities command in the device service. If any of the specified parameters MinResults and WaitTime exceed the supported range, a device shall adapt them instead of responding with an error.

REQUEST:

SearchToken [tt:JobToken]
Specifies the search session.

MinResults – optional [xs:int]
Specifies the minimum number of results that should be returned. If the total number of remaining event is lower than MinResults in a completed search, all remaining events should be returned.

MaxResults – optional [xs:int]
Specifies the maximum number of results to return.

WaitTime – optional [xs:duration]
Defines the maximum time to block, while waiting for results.

RESPONSE:

ResultList [tt: FindMetadataResultList]
A structure containing:

The state of the search when the result is returned. Indicates if there can be more results, or if the search is complete.

A FindMetadataResult structure for each found set of Metadata matching the search.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:InvalidToken
The search token is invalid.

ACCESS CLASS:

READ_MEDIA

6.15 EndSearch

EndSearch stops an ongoing search session, causing any blocking result request to return and the SearchToken to become invalid. If the search was interrupted before completion, the point in time that the search had reached shall be returned. If the search had not yet begun, the StartPoint shall be returned. Note that an error message will occur if the search session has been already completed before this request. If the search was completed, the original EndPoint supplied by the Find operation shall be returned. When issuing EndSearch on a FindRecordings request, the EndPoint is undefined and shall not be used since the FindRecordings request doesn't have a StartPoint/EndPoint. This operation is mandatory in order to support a device implementing the recording search service.

REQUEST:

SearchToken [tt:Jobtoken]
Specifies the search session.

RESPONSE:

EndPoint [xs:dateTime]
The point in time where the search was at when ended.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:InvalidToken
The search token is invalid.

ACCES CLASS:

READ_MEDIA

6.16 GetServiceCapabilities

The capabilities reflect optional functions and functionality of a service. The information is static and does not change during device operation. The following capabilites are available:

MetadataSearch Indication if the device supports generic search of recorded metadata

GeneralStartEvents Indicates support for general virtual property events in the FindEvents method

REQUEST:

This is an empty message.

RESPONSE:

Capabilities [tse:Capabilities]
Contains the requested service capabilities using a hierarchical XML capability structure.

ACCESS CLASS:

PRE_AUTH

6.17 Recording event descriptions

A device shall generate the following events with the corresponding event message descriptions. A device supporting the recording search service shall record these notification messages so that clients can use FindEvents to search for these messages. All recording events that are generated by the device and inserted into the recording history shall have a root topic of tns1:RecordingHistory.

The UtcTime attribute of the NotficationMessage (see 10.5.3 of IEC 60839-11-31:2016) of all recording events shall specify the actual time relating to recording regardless of the sending time of the event.

```
Topic: tns1:RecordingHistory/Recording/State
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken"
Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="IsRecording" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

This message is sent whenever a client starts or stops recording for a specific recording. At the start of recording, IsRecording shall be set to true. At the stopping of the recording, IsRecording shall be set to false.

```
Topic: tns1:RecordingHistory/Track/State
<tt:MessageDescription IsProperty="true">
```

```
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken"
Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescription Name="Track" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="IsDataPresent" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

This message signals when the data for a track is present. When the data becomes present, a message with IsDataPresent set to true shall be sent. When the data becomes unavailable, the message with IsDataPresent set to false shall be sent.

A device may generate the following events. If the device supports these events, it shall always automatically record these notification messages so that clients can always use FindEvent for these messages.

Topic: tns1:RecordingHistory/Track/VideoParameters

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="Recording"
Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescription Name="Track" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="VideoEncoding"
              Type="tt:VideoEncoding"/>
    <tt:SimpleItemDescription Name="VideoWidth" Type="xs:int"/>
    <tt:SimpleItemDescription Name="VideoHeight" Type="xs:int"/>
    <tt:ElementItemDescription                    Name="VideoRateControl"
              Type="tt:VideoRateControl"/>
  </tt:Data>
</tt:MessageDescription>
```

Topic: tns1:RecordingHistory/Track/AudioParameters

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="Recording"
Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescription Name="Track" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="AudioEncoding"
              Type="tt:AudioEncoding"/>
    <tt:SimpleItemDescription Name="AudioSampleRate" Type="xs:int"/>
    <tt:SimpleItemDescription Name="AudioBitrate" Type="xs:int"/>
  </tt:Data>
</tt:MessageDescription>
```

The device shall send either message (depending on the track data type) whenever any of these properties changes.

6.18 XPath dialect

This subclause defines a W3C XPATH 1.0 dialect that a device that realises the search service shall implement to parse the Xpath strings that are passed to the methods of the search service.

```
Dialect=http://www.onvif.org/ver10/tse/searchFilter

[1] Expression::= BoolExpr | Expression 'and' Expression
     | Expression 'or' Expression | '('Expression')' |
'not''('Expression')'
[2] BoolExpr::= 'boolean''('PathExpr')' | 'contains''(' ElementPath ',' '"'
String '"' ')'
[3] PathExpr::= '//SimpleItem' NodeTest | '//ElementItem' NodeTest |
     ElementTest
[4] NodeTest::= '['AttrExpr']'
[5] AttrExpr::= NameComp | ValueComp | AttrExpr 'and' AttrExpr | AttrExpr
'or' AttrExpr | 'not''(AttrExpr')'
[6] NameComp::= NameAttr'='"'String'"'
[7] ValueComp::= ValueAttr Operator '"'String'"'
[8] Operator::= '=' | '!=' | '<' | '<=' | '>' | '>='
[9] NameAttr::= '@Name'
[10] ValueAttr::= '@Value'
[11] ElementTest::= '/' ElementPath '['NodeComp']'
[12] ElementPath::= ElementName ElementName*
[13] ElementName::= '/' String
[14] NodeComp::= NodeName Operator '"' String '"'
[15] NodeName::= '@' String | String
```

Example of an XPath expression used to find recordings from the basement where there is at least one track containing video:

```
boolean(//Source[Location = "Basement"]) and
boolean(//Track[TrackType = "Video"])
```

7 Replay control

7.1 Request replay URI

GetReplayUri requests a URI that can be used to initiate playback of a recorded stream using RTSP as the control protocol. The URI is valid only as it is specified in the response. All implementations of the replay service shall support the GetReplayUri command.

REQUEST:

StreamSetup [tt:StreamSetup]
The StreamSetup element contains two parts. StreamType defines if a unicast or multicast media stream is requested. Transport specifies a chain of transport protocols defining the tunnelling of the media stream over different network protocols.

RecordingToken [tt:RefernceToken]
Indicates the recording to be streamed.

RESPONSE:

Uri [xs:anyURI]
Contains the URI to be used for requesting the media stream.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:InvalidStreamSetup
Specification of StreamType or Transport part in StreamSetup is not supported.

env:Sender – ter:OperationProhibited – ter:StreamConflict
Specification of StreamType or Transport part in StreamSetup causes conflict with other streams.

env:Sender – ter:InvalidArgVal – ter:NoRecording
The recording does not exist.

ACCESS CLASS:

READ_MEDIA

7.2 ReplayConfiguration

The ReplayConfiguration provides optional configuration information. Currently, it only contains the deprecated RTSP SessionTimeout, which can be controlled via the RTSP layer.

7.3 SetReplayConfiguration

SetReplayConfiguration changes the configuration of the replay service. The replay service shall allow its configuration to be changed using this command.

REQUEST:

Configuration [tt:ReplayConfiguration]
Holds the new configuration for the replay service.

RESPONSE:

This is an empty message.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:ConfigModify
The values in the configuration cannot be set.

ACCESS CLASS:

ACTUATE

7.4 GetReplayConfiguration

GetReplayConfiguration returns the current configuration of the replay service. The replay service shall allow its configuration to be retrieved using this command.

REQUEST:

This is an empty message.

RESPONSE:

Configuration [tt:ReplayConfiguration]
Holds the current configuration for the replay service.

FAULTS:

None

ACCESS CLASS:

READ_MEDIA

7.5 GetServiceCapabilities

The capabilities reflect optional functions and functionality of a service. The information is static and does not change during device operation. The following capabilities are available:

ReversePlayback Indicator that the device supports reverse playback as defined in 8.6.

SessionTimeoutRange Lists the upper and lower bound of the supported range for the session timeout. This capability defaults to the RTSP default value.

RTP_RTSP_TCP Indication if the device supports RTP/RTSP/TCP transport, see 10.2.1.4 of IEC 62676-2-31:2019.

RTSPOverWebSocket Indicates if the device supports RTSP/RTP Playback Streaming support over WebSocket and provides the WebSocket URI for replay as described in IEC 62676-2-31:2019, 10.2.1.6.

REQUEST:

This is an empty message.

RESPONSE:

Capabilities [trp:Capabilities]
Contains the requested service capabilities using a hierarchical XML capability structure.

FAULTS:

None

ACCESS CLASS:

PRE_AUTH

8 Playback

8.1 RTSP Usage

The replay protocol is based on RTSP (IETF RFC 2326). However, because RTSP does not directly support many of the features required by CCTV applications, this document defines several extensions to the protocol. These are detailed below.

This document makes the following stipulations on the usage of RTSP:

1) The server shall support RTP/RTSP/HTTP/TCP. This is the same transport protocol as a device that implements media streaming through the media service shall support, and the same requirements shall apply to replay streaming.
2) The server shall support the unicast RTP/UDP transport for streaming.
3) Clients should use a TCP-based transport for replay, in order to achieve reliable delivery of media packets.
4) The server may elect not to send RTCP packets during replay. In typical usage RTCP packets are not required, because usually a reliable transport will be used, and because absolute time information is sent within the stream, making the timing information in RTCP sender reports redundant.

8.2 RTSP describe

The SDP returned by the RTSP describe command shall include the TrackReference for each track of the recording to allow a client to map the tracks presented in the SDP to tracks of the recording. The tag shall use the following format:

a:x-onvif-track:<TrackReference>

For example:

```
NVS – NVT:     DESCRIBE rtsp://192.168.0.1 RTSP/1.0
               Cseq: 1
               User-Agent: ONVIF Rtsp client
               Accept: application/sdp

NVT – NVS:     RTSP/1.0 200 OK
               Cseq: 1
               Content-Type: application/sdp
               Content-Length: xxx

v=0
               o= 2890842807 IN IP4 192.168.0.1
                    m=video 0 RTP/AVP 26
                    a=control:rtsp://192.168.0.1/video
               a=x-onvif-track:VIDEO001
               m=audio 0 RTP/AVP 98
               a=control:rtsp://192.168.0.1/audio
               a=x-onvif-track:AUDIO001
```

8.3 RTP header extension

8.3.1 General

In order to allow clients to report a stable and accurate timestamp for each frame played back regardless of the direction of playback, it is necessary to associate an absolute timestamp with each packet, or each group of packets with the same RTP timestamp (e.g. a video frame). This is achieved using an RTP header extension containing an NTP timestamp and some additional information also useful for replay.

The replay mechanism uses the extension ID 0xABAC for the replay extension.

Table 3 shows the general form of an RTP packet containing this extension.

Table 3 – RTP packet layout

<table>
<tr><td colspan="2">V=2</td><td>P</td><td>X=1</td><td colspan="2">CC</td><td>M</td><td colspan="2">PT</td><td colspan="2">sequence number</td></tr>
<tr><td colspan="10">timestamp</td></tr>
<tr><td colspan="10">synchronization source (SSRC) identifier</td></tr>
<tr><td colspan="8">0xABAC</td><td colspan="2">length=3</td></tr>
<tr><td colspan="10">NTP timestamp…</td></tr>
<tr><td colspan="10">…NTP timestamp</td></tr>
<tr><td>C</td><td>E</td><td>D</td><td colspan="3">mbz</td><td colspan="3">Cseq</td><td colspan="2">padding</td></tr>
<tr><td colspan="10">payload…</td></tr>
</table>

The fields of this extension are as follows:

– NTP timestamp. An NTP (IETF RFC 1305) timestamp indicating the absolute UTC time associated with the access unit.

– C: 1 bit. Indicates that this access unit is a synchronization point or "clean point", e.g. the start of an intra-coded frame in the case of video streams.

– E: 1 bit. Indicates the end of a contiguous section of recording. The last access unit in each track before a recording gap, or at the end of available footage, shall have this bit set. When replaying in reverse, the E flag shall be set on the last frame at the end of the contiguous section of recording.

– D: 1 bit. Indicates that this access unit follows a discontinuity in transmission. It is primarily used during reverse replay; the first packet of each GOP has the D bit set since it does not chronologically follow the previous packet in the data stream (see 8.6).

– mbz: This field is reserved for future use and shall be zero.

– Cseq: 1 byte. This is the low-order byte of the CSeq value used in the RTSP PLAY command that was used to initiate transmission. When a client sends multiple, consecutive PLAY commands, this value may be used to determine where the data from each new PLAY command begins.

The replay header extension shall be present in the first packet of every access unit (for example, a video frame).

8.3.2 NTP timestamps

The NTP timestamps in the RTP extension header shall increase monotonically over successive packets within a single RTP stream. They should correspond to wall-clock time as measured at the original transmitter of the stream, adjusted if necessary to preserve monotonicity.

8.3.3 Compatibility with the JPEG header extension

The replay header extension may co-exist with the header extension used by the JPEG RTP profile; this is necessary to allow replay of JPEG streams that use this extension. The JPEG extension is simply appended to the replay extension; its presence is indicated by an RTP header extension length field with a value greater than 3, and by the extension start codes of 0xFFD8 or 0xFFFF at the start of the fourth word of the extension content.

Table 4 illustrates a JPEG packet that uses both extensions:

Table 4 – RTP packet with JPEG header layout

<table>
<tr><td colspan="2">V=2</td><td>P</td><td>X=1</td><td colspan="2">CC</td><td>M</td><td colspan="2">PT</td><td colspan="2">sequence number</td></tr>
<tr><td colspan="10">timestamp</td></tr>
<tr><td colspan="10">synchronization source (SSRC) identifier</td></tr>
<tr><td colspan="7">0xABAC</td><td colspan="3">length=N+4</td></tr>
<tr><td colspan="10">NTP timestamp…</td></tr>
<tr><td colspan="10">…NTP timestamp</td></tr>
<tr><td>C</td><td>E</td><td>D</td><td colspan="3">mbz</td><td colspan="3">CSeq</td><td colspan="2">padding</td></tr>
<tr><td colspan="7">0xFFD8</td><td colspan="3">jpeglength=N</td></tr>
<tr><td colspan="10">extension payload: sequence of additional JPEG marker segments padded with 0xFF to the total extension length</td></tr>
<tr><td colspan="10">payload…</td></tr>
</table>

8.4 RTSP feature tag

The replay service uses the "onvif-replay" feature tag to indicate that it supports the RTSP extensions described in this document.

Example:

```
C->S:   SETUP rtsp://server.com/foo/bar/baz.rm RTSP/1.0
        Cseq: 302
        Require: onvif-replay

S->C:   RTSP/1.0 551 Option not supported
        Cseq: 302
        Unsupported: onvif-replay
```

The replay server shall accept a SETUP and PLAY command that includes a Require header containing the onvif-replay feature tag.

8.5 Initiating playback

8.5.1 General

Playback is initiated by means of the RTSP PLAY method. For example:

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
Cseq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-
Rate-Control: no
```

The ReversePlayback capability, defined in 7.5, signals if a device supports reverse playback. Reverse playback is indicated using the Scale header field with a negative value. For example, to play in reverse without no data loss, a value of −1,0 would be used.

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
Cseq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-
Rate-Control: no
Scale: -1.0
```

If a device supports reverse playback, it shall accept a Scale header with a value of −1,0. A device may accept other values for the Scale parameter. Unless the Rate-Control header is set to "no" (see below), the Scale parameter is used in the manner described in IETF RFC 2326. If Rate-Control is set to "no", the Scale parameter, if it is present, shall be either 1,0 or −1,0, to indicate forward or reverse playback respectively. If it is not present, forward playback is assumed.

8.5.2 Range header field

A device shall support the Range field expressed using absolute times as defined by IETF RFC 2326. Absolute times are expressed using the utc-range from IETF RFC 2326.

Either open or closed ranges can be used. In the case of a closed range, the range increases (end time later than start time) for forward playback and decreases for reverse playback. The direction of the range shall correspond to the value of the Scale header.

In all cases, the first point of the range indicates the starting point for replay.

The time itself shall be given as

```
utc-range = "clock" ["=" utc-range-spec]
utc-range-spec = ( utc-time "-" [ utc-time ] ) / ( "-" utc-time )
utc-time = utc-date "T" utc-clock "Z"
utc-date = 8DIGIT
utc-clock = 6DIGIT [ "." 1*9DIGIT ]
```

as defined in IETF RFC 2326.

Examples:

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
Cseq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-20090615T115000Z
Rate-Control: no

PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
Cseq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T115000.440Z-20090615T114900Z
Rate-Control: no
Scale: -1.0
```

8.5.3 Rate-Control header field

This specification introduces the Rate-Control header field, which can be either "yes" or "no". If the field is not present, "yes" is assumed, and the stream is delivered in real time using standard RTP timing mechanisms. If this field is "no", the stream is delivered as fast as possible, using only the flow control provided by the transport to limit the delivery rate.

The important difference between these two modes is that with "Rate-Control=yes", the server is in control of the playback speed, whereas with "Rate-Control=no" the client is in control of the playback speed.

When replaying multiple tracks of a single recording, started by a single RTSP PLAY command and not using rate-control, the data from the tracks should be multiplexed in time in the same order as they were recorded.

A device shall support operation with "Rate-Control=no" for playback.

8.5.4 Frames header field

The Frames header field may be used to reduce the number of frames that are transmitted, for example to lower bandwidth or processing load. Three modes are possible:

1) Intra frames only. This is indicated using the value "intra", optionally followed by a minimum interval between successive intra frames in the stream. The latter can be used to limit the number of frames received even in the presence of "I-frame storms" caused by many receivers requesting frequent I-frames.
2) Intra frames and predicted frames only. This is indicated using the value "predicted". This value can be used to eliminate B-frames if the stream includes them.
3) All frames. This is the default.

Examples:

To request intra frames only:

```
Frames: intra
```

To request intra frames with a minimum interval of 4 000 milliseconds:

```
Frames: intra/4000
```

To request intra frames and predicted frames only:

```
Frames: predicted
```

To request all frames (note that it is not necessary to explicitly specify this mode but the example is included for completeness):

```
Frames: all
```

The interval argument used with the "intra" option refers to the recording timeline, not playback time; thus for any given interval the same frames are played regardless of playback speed. The interval argument shall not be present unless the Frames option is "intra".

The server shall support the Frames header field. This does not preclude the use of the Scale header field as an alternative means of limiting the data rate. The implementation of the Scale header field may vary between different server implementations, as stated by IETF RFC 2326.

A device shall support the Frames parameters "intra" and "all" for playback.

8.5.5 Synchronization points

The transmitted video stream shall begin at a synchronization point, see 6.8 of IEC 62676-11-31:2019. The rules for choosing the starting frame are as follows:

– If the requested start time is within a section of recorded footage, the stream starts with the first clean point at or before the requested start time. This is the case regardless of playback direction.

– If the requested start time is within a gap in recorded footage and playback is being initiated in the forwards direction, the stream starts with the first clean point in the section following the requested start time.

– If the requested start time is within a gap in recorded footage and playback is being initiated in the reverse direction, the stream starts with the last clean point in the section preceding the requested start time.

8.6 Reverse replay

8.6.1 Initiation

Reverse replay is initiated using the Scale header field with a negative value as described above.

8.6.2 Packet transmission order

The example in Figure 7 shows how frames are transmitted during normal forward playback for a recording with two short video clips each consisting of two GOPs. As shown in Figure 7, all packets are transmitted in recording order. The last frame before a gap is marked with the E flag to signal a gap in the recording.

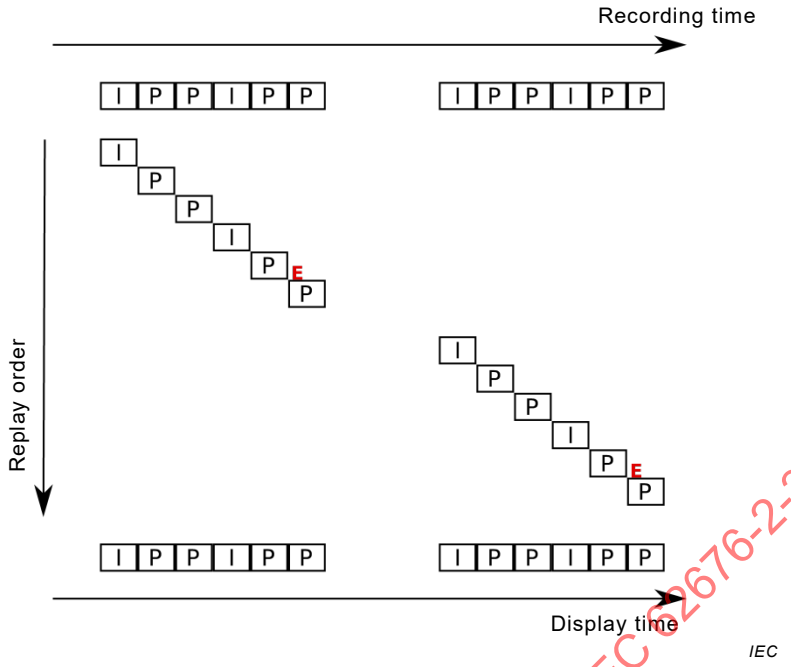

Figure 7 – Packet transmission during forward playback

The order in which video packets are transmitted during reverse replay is based on GOPs, where a GOP consists of a clean point followed by a sequence of non-cleanpoint packets.

During reverse playback, GOPs shall be sent in reverse order, but packets within a GOP shall be sent in forward order. The first packet of each GOP shall have the "discontinuity" bit set in its RTP extension header. The last frame of a GOP immediately following a gap (or the beginning of available footage) shall have the E bit set in its RTP extension header.

When transmitting only key frames, or when the codec is not motion-based (e.g. JPEG), a GOP is considered to consist of a single frame, but can still be composed of multiple packets. In this case, the packets within each frame shall be again sent in forward order, while the frames themselves shall be sent in reverse order.

Audio and metadata streams may be transmitted in an order mirroring that of the video stream. Thus packets from these streams are sent in forward playback order until the occurrence of a packet (generally a video packet) with the D bit set in the extension header, at which point they jump back to a point before the discontinuity.

Note that reverse playback of an audio packet isn't useful. Therefore, audio packets should generally not be transmitted during reverse playback.

The example of Figure 8 shows, for the same recording as depicted in Figure 7, how packets are transmitted during reverse forward playback. As shown in Figure 8, all packets are transmitted in recording order.

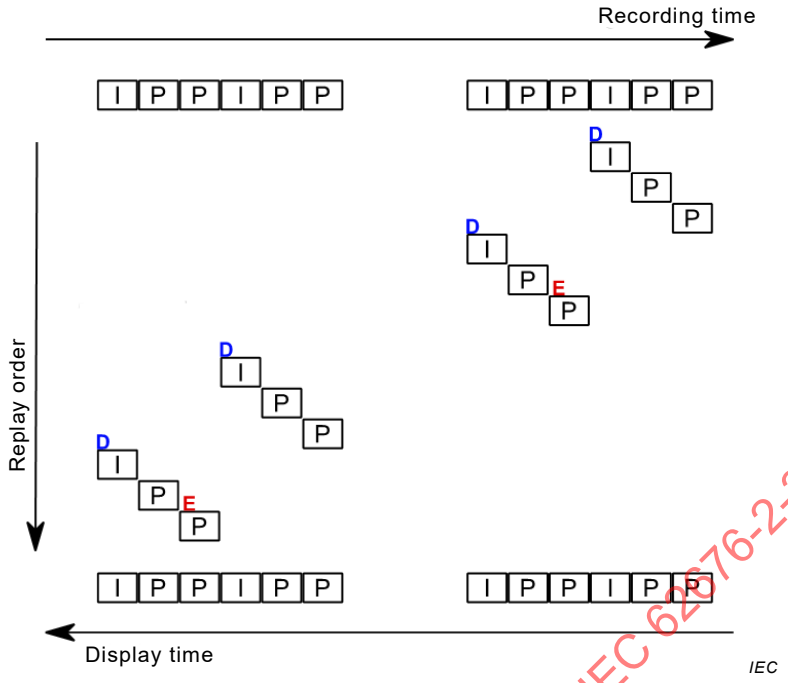

Figure 8 – Packet transmission during reverse playback

8.6.3 RTP sequence numbers

The RTP sequence numbers of packets transmitted during reverse playback shall increment monotonically in the order of delivery, not in the intended order of playback.

8.6.4 RTP timestamps

The use of RTP timestamps depends on the value of the Rate-Control header. If the value of this header is "no" (i.e. the client controls playback speed), the RTP timestamps are derived from the original sampling times of the recorded frames. If the Rate-Control header is not present or has the value "yes" (i.e. the server controls playback speed), the RTP timestamps correspond to playback timing as described in IETF RFC 2326, Appendix B.

If Rate-Control is "no" the RTP timestamps of packets transmitted during reverse playback shall be the same as they would be if those same packets were being transmitted in the forwards direction. Unlike the sequence numbers, the RTP timestamps correspond to the original recording order, not the delivery order. The server may use the same RTP timestamps that were originally received when the stream was recorded.

This means that successive RTP packets of a single GOP will always have increasing RTP timestamps (see transmission order above), but that the timestamp on index frames of successively received GOPs will decrease during reverse replay.

If Rate-Control is "yes", the RTP timestamps of packets transmitted during reverse playback shall indicate the times at which each frame should be rendered at the client. Thus, successive packets of a single GOP will have decreasing RTP timestamps (since the first one delivered should be played last), and the timestamps on index frames will increase. In this mode, the interval between successive timestamps depends on the values of the Speed and Scale headers, as described in IETF RFC 2326, Appendix B

8.7 RTSP Keepalive

When rate control is disabled and the RTP stream is tunnelled through the RTSP connection (i.e. using the RTP/RTSP/TCP or RTP/RTSP/HTTP/TCP transports), the client shall be aware that it may not be able to receive the response to any request if, for example, replay is paused.

8.8 Currently recording footage

If the client commences playback from the current real world time or shortly before it, it can end up playing footage in real time as it is being recorded. In this event, the server simply continues to send stream data to the client as it receives it.

Note that the E bit is not set on access units currently being recorded, even though each access unit sent to the replay client will typically be the last one known to the server. If recording stops, however, the E bit is set on the last access unit of the recording.

8.9 End of footage

If playback reaches a point after which there is no further data in one or more of the streams being sent, it stops transmitting data but does not enter the "paused" state. If the server resumes recording after this has happened, delivery will resume with the new data as it is received.

8.10 Go To Time

As stated in 10.5 of IETF RFC 2326, a PLAY command received when replay is already in progress will not take effect until the existing play operation has completed. This specification adds a new RTSP header, "Immediate", which overrides this behaviour for the PLAY command that it is used with:

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
CSeq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-
Rate-Control: no
Immediate: yes
```

If the server receives a PLAY command with the Immediate header set to "yes", it will immediately start playing from the new location, cancelling any existing PLAY command. The first packet sent from the new location shall have the D (discontinuity) bit set in its RTP extension header.

A device shall support the immediate header field for playback with the value "yes". The behaviour without immediate header or value "no" is not defined by the specification.

8.11 Use of RTCP

A server is not required to send RTCP packets. If it does send them, the following rules apply:

- If Rate Control is enabled (see 8.5.3), RTCP packets shall be constructed and transmitted as specified in IETF RFC 3550. In particular, the NTP timestamp in a sender report indicates the current wallclock time, and is not related to the NTP timestamps embedded in the RTP extension headers in the data streams.
- If Rate Control is not enabled, both the NTP timestamp and RTP timestamp in each sender report shall be set to zero.

9 Export file format

9.1 Required side information

SurveillanceExportBox is required. It is recommended that SurveillanceExportBox be placed as early as possible in files, for maximum utility.

In order to be able to associate the recording with a camera/microphone and the exporting system the following information shall be placed in the box:

- Source – Description of the video source
  - Name – Name of the camera
  - URL – Address under which the camera can be accessed
  - MAC – Unique physical address of the camera (examples: 08-00-27-00-0C-15, 08:00:27:00:0C:15, 080027000C15)
  - Line – Input line number token for multi channel devices
- Source – Description of the audio source
  - Name – Name of the microphone
  - URL – Address under which the microphone can be accessed
  - MAC – Unique physical address of the microphone
- Export – Unit executing the export
  - Name – Name of the exporting unit
  - URL – Address under which the exporting unit can be accessed
  - MAC – Unique physical address of the exporting unit
  - Time – Date and time information on when the export was executed (start time)
  - Operator – Name or identification of the operator performing the export

SurveillanceExportBox

Box Type:	‘suep’
Container:	Meta Box (‘meta’), file level
Mandatory:	Yes
Quantity:	Exactly one

This box shall provide ordered information about source and export units. Media tracks are identified by their respective UUID. Information has to be available only for the type of media present but separately for each and every track. Information about the exporting unit is leading.

Syntax

```
class SurveillanceExportBox

      extends      FullBox('suep', version = 0, 0){
      string       ExportUnitName;
      string       ExportUnitURL;
      string       ExportUnitMAC;
      UInt(64)     ExportUnitTime;
      string       ExportOperator

      Uint(8)      video_count;
     int i;
     for (i=0; i < video_count; i++) {

            UInt(128)       VideoTrackUUID
            string          VideoSourceName;
            string          VideoSourceURL;
            string          VideoSourceMAC;
            string          VideoSourceLine;
      }

      Uint(8)  audio_count;
      int j;

     for (j=0; j < audio_count; j++) {

            UInt(128)       AudioTrackUUID
            string          AudioSourceName;
            string          AudioSourceURL;
            string          AudioSourceMAC;
      }
}
```

Semantics

String items are null-terminated strings in UTF-8 characters. If not applicable, the string shall contain the null-termination only.

`ExportOperator` is a string that gives the name or identification of the operator performing the export. This string may be empty.

`ExportUnitTime` is an integer that gives date and time designation as defined in ISO/IEC 14496-12 of when the export operation has been started.

`video_count` is an integer that gives the number of entries in the following list and equals the total number of video tracks.

`audio_count` is an integer that gives the number of entries in the following list and equals the total number of audio tracks.

9.2 Timing

The `startTime` element of `AFIdentificationBox`[3] shall contain the UTC based time of the first media sample in the fragment.

9.3 Correction of start time

CorrectStartTimeBox

Box Types: 'cstb

Container: Protection Scheme Information Box (‘sinf’)

Mandatory: No

Quantity: Zero or one per signing instance

Syntax

```
aligned(8) class CorrectStartTimeBox extends Box ('cstb') {
      UInt(32) entry_count;
      for (i=0; i < entry_count; i++) {
            unsigned int(32) track_ID;
            unsigned int(64) startTime;
      }
)
```

Semantics

`track_ID` is an integer that provides a reference to another track in the presentation. track_IDs are never re-used and cannot be equal to zero.

`startTime` the UTC based time represented by the number of 100-nanosecond intervals since January 1, 1601 of the first media sample in the first fragment.

9.4 Signature

9.4.1 Preparing the signature input

The input to the signature algorithm are all boxes of the file including boxes for the signature to be created, with the corresponding signature string set to zero. Particularly, the input contains signatures that are already present for repeated signing operations.

9.4.2 Generating the signature

Implementations of this specification shall support RSASSA-PSS signatures as specified in ISO/IEC 14888-2 and PKCS#1 v2.1 with

- SHA-256 as specified in FIPS 180-4 as cryptographic hash function;
- an RSA modulus length of 2 048 bits;
- MGF1 as specified in PKCS#1 v2.1 as mask generation algorithm with SHA-256 as cryptographic hash function;
- Salt length 20;
- Trailer field number as specified by the trailerFieldBC constant.

Implementations may support other digital signature algorithms, if appropriate.

---

3 Box definitions can be found in ISO/IEC 23000-10.

The generated signature string has to be included in SignatureBox as defined in 9.4.3.

Generating and maintaining parameters of the signature algorithm, particularly signature and verification keys, is outside the scope of this document. Recommendations given, for example, in FIPS 186-4 should be followed where appropriate.

9.4.3 Include the generated signature in the file

There are no changes to the file itself or the content after the signing operation has been performed. The sole exception is the input of the signature at the appropriate place.

The following box definitions provide for signature identification and inclusion. Encryption is not required; therefore OriginalFormatBox is not necessary.

Item Protection Box[4]

Box types: ‘ipro’
Container: Meta box (‘meta’)
Mandatory: Yes
Quantity: Exactly one

The `protection_count` shall be 1.

Protection Scheme Info Box[4]

Box Types: ‘sinf’
Container: Item Protection Box (‘ipro’)
Mandatory: Yes
Quantity: One per signing instance

Contains exactly one SchemeTypeBox and exactly one SchemeInformationBox.

Scheme Type Box[4]

Box Types: ‘schm’
Container: Protection Scheme Information Box (‘sinf’)
Mandatory: Yes
Quantity: One per signing instance

The `scheme_type` shall be 0x6F656666 ("Onvif Export File Format").

The `scheme_version` shall be 0x00010000 (version 1).

Scheme Information Box[4]

Box Types: ‘schi’
Container: Protection Scheme Information Box (‘sinf’)
Mandatory: Yes
Quantity: One per signing instance

4 Box definitions can be found in ISO/IEC 14496-12.

Contains exactly one SignatureBoxand exactly one CertificateBox. May also contain exactly one AdditionalUserInformationBox and exactly one SignatureConfigurationBox.

SignatureBox

Box Types: ‘sibo’

Container: Scheme Information Box (‘schi’)

Mandatory: Yes

Quantity: One per signing instance

Syntax

```
aligned(8) class SignatureBox
      extends Box('sibo') {
     bit(8)  signature[];
}
```

Semantics

`signature` binary byte array. Length depends on used RSA key length.

CertificateBox

Box Type: ‘cert’

Container: Scheme Information Box (‘schi’)

Mandatory: Yes

Quantity: One per signing instance

Syntax

```
aligned(8) class CertificateBox
      extends Box('cert') {
      bit(8) data[];
}
```

Semantics

data is the DER encoded binary byte array representation of the certificate for the key that should be used to verify the signature in the SignatureBox

9.5 Repeated signing

To add, for example, electronic receiving stamps, repeated signing of the file might be applied.

Repeat steps defined in 9.4.1 and 9.4.2 and append another ProtectionSchemeInfoBox at the foot of the list of already existing boxes of that type as defined in 9.4.2, while not changing `protection_count` in the ItemProtectionBox. Parsers are required to check for the existence of multiple ProtectionSchemeInfoBox despite `protection_count` being fixed at 1, because any change of content that has already been signed would render the appropriate signature invalid. An optional AdditionalUserInformationBox can be used in order to add information.

Only if the signature algorithm applied deviates from the default shall the ‘sigC’ box be present.

See Annex A for more information.

SignatureConfigurationBox

Box Types: ‘sigC’

Container: Scheme Information Box (‘schi’)

Mandatory: No

Quantity: Zero or one per signing instance

Syntax

```
aligned(8) class SignatureConfigurationBox
      extends Box('sigC') {

      bit(8)AlgorithmIdentifier[];
}
```

Semantics

`AlgorithmIdentifier` is the signature algorithm identifier with optional parameters as defined by IETF RFC 3280 and IETF RFC 4055. It is encoded using the ASN.1 distinguished encoding rules (DER) and has the structure:

```
AlgorithmIdentifier::=  SEQUENCE  {
       algorithm               OBJECT IDENTIFIER,
       parameters              ANY DEFINED BY algorithm OPTIONAL  }
```

In order to include optional user information data related to an additional signature, the ‘auib’ box is provided.

AdditionalUserInformationBox

Box Types: ‘auib’

Container: Scheme Information Box (‘schi’)

Mandatory: No

Quantity: Zero or one per signing instance

Syntax

```
aligned(8) class AdditionalUserInformationBox
      extends Box('auib') {
      string  UserInformation;
}
```

Semantics

`UserInformation` is a null terminated string in UTF-8 characters.

10 Receiver service

10.1 General

This service offers commands to manage Receiver objects, which are used to receive media streams from other devices. A Receiver object contains the information how to setup the stream, the mode of the receiver and the stream URI (MediaUri). A device implementing the receiver service shall at least support media URIs of 128 octets in length. The Receiver – MaximumRTSPURILength capability indicates the maximum length supported by the device.

The IP or DNS address in the transmit URI given to the receiver, is the address that the device hosting the receiver service will use to access the transmit device. If, for example, the client has to communicate through a NAT router to access the transmitter and the receiver,

the transmitter address that the client gives the receiver (in this case a local network address) may not be the same address that the client would use to access the transmitter (in this case an external network address).

A device implementing the receiver service shall support RTP transfer via UDP and RTP transfer via RTSP/HTTP/TCP, see IEC 62676-11-31. A device may support other RTP transport protocols and shall indicate what it supports with the appropriate capability (see 10.6).

10.2 Synchronization points

Because receivers use RTSP addresses to specify the source of the stream, they do not necessarily have access to the web services interface of the transmitter. This means that they cannot use the SetSynchronizationPoint command described in 6.8 of IEC 62676-11-31:2019.

Instead, receivers should use the PLI message described in IETF RFC 4585 to request a synchronization point.

10.3 Persistence

All the objects created within the receiver service shall be persistent – i.e. they shall survive a power cycle. Likewise, all the configuration data in the objects shall be persistent.

10.4 Receiver modes

A receiver can operate in three distinct modes:

AlwaysConnect: the receiver attempts to maintain a persistent connection to the configured endpoint.

NeverConnect: the receiver does not attempt to connect.

AutoConnect: the receiver connects on demand, as required by consumers of the media streams.

10.5 Receiver commands

10.5.1 GetReceivers

This operation lists all receivers that currently exist on the device. A device implementing the receiver service shall support this command.

REQUEST:

This is an empty message.

RESPONSE:

Receivers – optional, unbounded [tt:Receiver]
A list of receivers.

FAULTS:

None

ACCESS CLASS:

READ MEDIA

10.5.2 GetReceiver

This operation retrieves the details of a specific receiver whose token is known to the client. A device implementing the receiver service shall support this command.

REQUEST:

ReceiverToken [tt:ReferenceToken]
The token of the requested receiver.

RESPONSE:

Receiver [tt:Receiver]
The details of the requested receiver.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:UnknownToken
The receiver indicated by ReceiverToken does not exist.

ACCESS CLASS:

READ MEDIA

10.5.3 CreateReceiver

This operation creates a new receiver. A device implementing the receiver service shall support this command.

REQUEST:

Configuration [tt:ReceiverConfiguration]
The initial configuration of the receiver.

RESPONSE:

Receiver [tt:Receiver]
The details of the receiver that was created.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
The specified configuration is invalid.

env:Receiver – ter:Action – ter:MaxReceivers
The maximum supported number of receivers has been reached.

ACCESS CLASS:

ACTUATE

10.5.4 DeleteReceiver

This operation deletes an existing receiver. A device may reject deletion of a receiver which is in use. A device implementing the receiver service shall support this command.

REQUEST:

ReceiverToken [tt:ReferenceToken]

The token of the receiver to be deleated.

RESPONSE:

This is an empty message.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:UnknownToken
The receiver indicated by ReceiverToken does not exist.

env: Receiver – ter:Action – ter:CannotDeleteReceiver
It is not possible to delete the specified receiver, for example because it is currently in use.

ACCESS CLASS:

ACTUATE

10.5.5 ConfigureReceiver

This operation configures a receiver. A device implementing the receiver service shall support this command.

REQUEST:

ReceiverToken [tt:ReferenceToken]
The token of the requested receiver.

Configuration [tt:ReceiverConfiguration]
The new configuration for the receiver.

RESPONSE:

This is an empty message.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:UnknownToken
The receiver indicated by ReceiverToken does not exist.

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
The specified configuration is invalid.

ACCESS CLASS:

ACTUATE

10.5.6 SetReceiverMode

This operation may be used to set the mode of the receiver independently of the rest of its configuration. A device implementing the receiver service shall support this command.

REQUEST:

ReceiverToken [tt:ReferenceToken]
The token of the requested receiver.

ReceiverMode [tt:ReceiverMode]
The new mode of the receiver.

RESPONSE:

This is an empty message.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:UnknownToken
The receiver indicated by ReceiverToken does not exist.

ACCESS CLASS:

ACTUATE

10.5.7 GetReceiverState

This operation determines whether the receiver is currently disconnected, connected or attempting to connect. A device implementing the receiver service shall support this command.

REQUEST:

ReceiverToken [tt:ReferenceToken]
The token of the requested receiver.

RESPONSE:

State [tt:ReceiverState]
The current state of the receiver.

FAULTS:

env:Sender – ter:InvalidArgVal – ter:UnknownToken
The receiver indicated by ReceiverToken does not exist.

ACCESS CLASS:

READ MEDIA

10.6 GetServiceCapabilitites

The capabilities reflect optional functions and functionality of a service. The information is static and does not change during device operation. The following capabilites are available:

RTP_Multicast: Indication if the device supports receiving of RTP Multicast.

RTP_TCP: Indication if the device supports receiving of RTP over TCP.

RTP_RTSP_TCP: Indication if the device supports receiving of RTP over RTSP over TCP.

SupportedReceivers: The maximum number of receivers the device supports..

MaximumRTSPURILength: The maximum length allowed for RTSP URIs.

REQUEST:

This is an empty message.

RESPONSE:

Capabilities [trv:Capabilities]
List of capabilities as defined above.

FAULTS:

None

ACCESS CLASS:

PRE_AUTH

10.7 Events

10.7.1 General

A device implementing the receiver service shall dispatch events through the event service. It shall be capable of generating the events listed in this chapter whenever the condition that fires the event occurs.

10.7.2 ChangeState

Whenever a receiver changes state, the device shall dispatch the following event:

```
Topic: tns1: Receiver/ChangeState
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="ReceiverToken" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="NewState" Type="tt:ReceiverState"/>
    <tt:SimpleItemDescription Name="MediaUri" Type="tt:MediaUri" minOccurs="0"/>
  </tt:Data>
</tt:MessageDescription>
```

10.7.3 Connection Failed

If a receiver fails to establish a connection, the device shall dispatch the following event:

```
Topic: tns1: Receiver/ConnectionFailed
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="ReceiverToken" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="MediaUri" Type="tt:MediaUri"/>
  </tt:Data>
</tt:MessageDescription>
```

Annex A
(informative)

Repeated signing

Figure A.1 shows the characterization of the box arrangement defined in the present specification for data export.

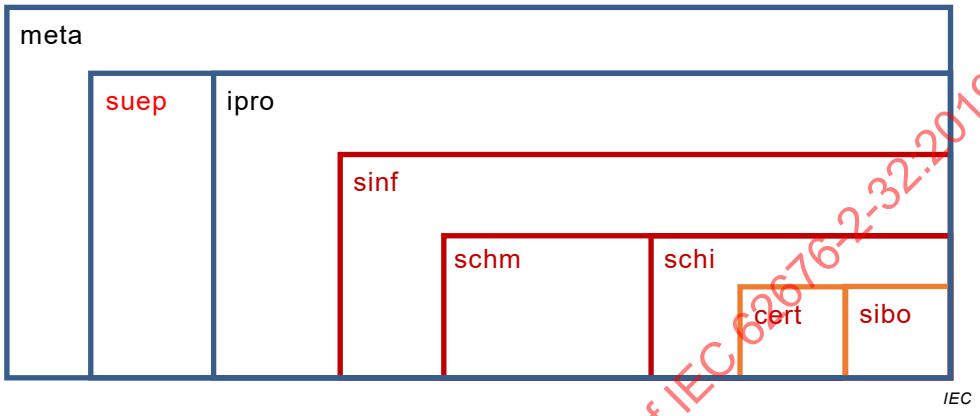

Figure A.1 – Single signature box arrangement

Figure A.2 shows the characterization of the box arrangement after repeated signing.

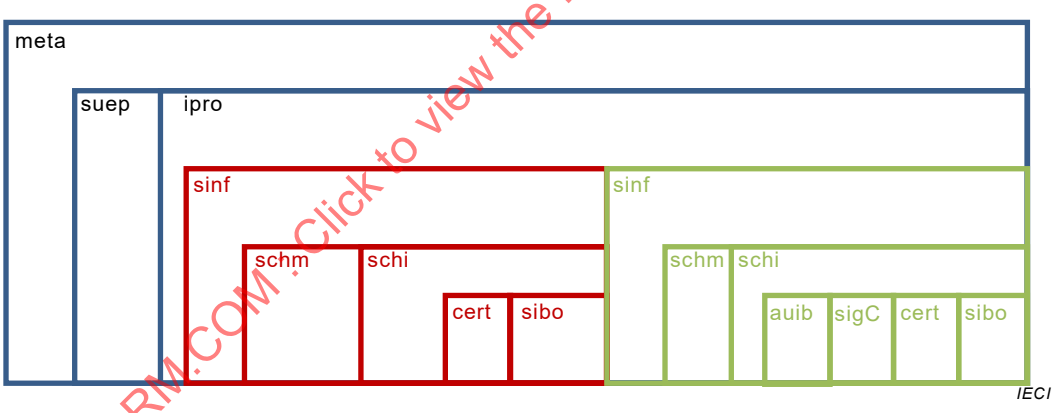

Figure A.2 – Repeated signature box arrangement

The red colour represents the signature introduced at the export stage. The green color represents another signing operation happening after the export stage. The CertificateBox provides the public key for signature verification. The SignatureConfigurationBox information contains information describing a differing signature algorithm and its parameters. Additional information has been added in the AdditionalUserInformationBox.

In order to check validity of a signature, the signature itself has to be taken from SignatureBox and the bit values for the signature string be set to zero. The hashing operation is performed followed by the signature operation on the hash value. Now the two signatures can be compared.

Example: check validity of the first (red) signature from above

- Remove the (green) boxes created for the second signing
- Re-adjust box sizes of ‘ipro’ and ‘meta’ according to the size of removed ‘sinf’ box

- Read the public key from the red CertificateBox (do not change the box content)
- Take out the signature from the red SignatureBox
- Set the bit values of the red SignatureBox to zero
- Perform hash operation on the remaining file data
- Perform signature operation on the obtained hash value
- Compare the just generated signature with the signature taken out before

Annex B
(normative)

Schema files

B.1 Recording control

```
<?xml version="1.0" encoding="utf-8"?><?xml-stylesheet type="text/xsl"
href="../ver20/util/onvif-wsdl-viewer.xsl"?><wsdl:definitions
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:trc="http://www.onvif.org/ver10/recording/wsdl"
targetNamespace="http://www.onvif.org/ver10/recording/wsdl">
   <wsdl:types>
      <xs:schema xmlns:tt="http://www.onvif.org/ver10/schema"
targetNamespace="http://www.onvif.org/ver10/recording/wsdl"
elementFormDefault="qualified" version="16.12">
         <xs:import namespace="http://www.onvif.org/ver10/schema"
schemaLocation="../ver10/schema/onvif.xsd"/>
         <xs:element name="GetServiceCapabilities">
            <xs:complexType>
               <xs:sequence/>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetServiceCapabilitiesResponse">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="Capabilities" type="trc:Capabilities"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:complexType name="Capabilities">
            <xs:sequence>
               <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="DynamicRecordings" type="xs:boolean"/>
            <xs:attribute name="DynamicTracks" type="xs:boolean"/>
            <xs:attribute name="Encoding" type="trc:EncodingTypes"/>
            <xs:attribute name="MaxRate" type="xs:float"/>
            <xs:attribute name="MaxTotalRate" type="xs:float"/>
            <xs:attribute name="MaxRecordings" type="xs:float"/>
            <xs:attribute name="MaxRecordingJobs" type="xs:int"/>
            <xs:attribute name="Options" type="xs:boolean"/>
            <xs:attribute name="MetadataRecording" type="xs:boolean"/>
            <xs:attribute name="SupportedExportFileFormats" type="tt:StringAttrList"/>
            <xs:anyAttribute processContents="lax"/>
         </xs:complexType>
         <xs:simpleType name="EncodingTypes">
            <xs:list itemType="xs:string"/>
         </xs:simpleType>
         <xs:element name="Capabilities" type="trc:Capabilities"/>
         <xs:element name="CreateRecording">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="RecordingConfiguration"
type="tt:RecordingConfiguration"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="CreateRecordingResponse">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="RecordingToken" type="tt:RecordingReference"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="DeleteRecording">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="RecordingToken" type="tt:RecordingReference"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
```

```
            <xs:element name="DeleteRecordingResponse">
               <xs:complexType>
                  <xs:sequence/>
               </xs:complexType>
            </xs:element>
            <xs:element name="GetRecordings">
               <xs:complexType>
                  <xs:sequence/>
               </xs:complexType>
            </xs:element>
            <xs:element name="GetRecordingsResponse">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="RecordingItem" type="tt:GetRecordingsResponseItem"
minOccurs="0" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="SetRecordingConfiguration">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="RecordingToken" type="tt:RecordingReference"/>
                     <xs:element name="RecordingConfiguration"
type="tt:RecordingConfiguration"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="SetRecordingConfigurationResponse">
               <xs:complexType>
                  <xs:sequence/>
               </xs:complexType>
            </xs:element>
            <xs:element name="GetRecordingConfiguration">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="RecordingToken" type="tt:RecordingReference"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="GetRecordingConfigurationResponse">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="RecordingConfiguration"
type="tt:RecordingConfiguration"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="CreateTrack">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="RecordingToken" type="tt:RecordingReference"/>
                     <xs:element name="TrackConfiguration" type="tt:TrackConfiguration"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="CreateTrackResponse">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="TrackToken" type="tt:TrackReference"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="DeleteTrack">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="RecordingToken" type="tt:RecordingReference"/>
                     <xs:element name="TrackToken" type="tt:TrackReference"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="DeleteTrackResponse">
               <xs:complexType>
                  <xs:sequence/>
               </xs:complexType>
            </xs:element>
            <xs:element name="GetTrackConfiguration">
               <xs:complexType>
                  <xs:sequence>
```

```
                <xs:element name="RecordingToken" type="tt:RecordingReference"/>
                <xs:element name="TrackToken" type="tt:TrackReference"/>
             </xs:sequence>
          </xs:complexType>
       </xs:element>
       <xs:element name="GetTrackConfigurationResponse">
          <xs:complexType>
             <xs:sequence>
                <xs:element name="TrackConfiguration" type="tt:TrackConfiguration"/>
             </xs:sequence>
          </xs:complexType>
       </xs:element>
       <xs:element name="SetTrackConfiguration">
          <xs:complexType>
             <xs:sequence>
                <xs:element name="RecordingToken" type="tt:RecordingReference"/>
                <xs:element name="TrackToken" type="tt:TrackReference"/>
                <xs:element name="TrackConfiguration" type="tt:TrackConfiguration"/>
             </xs:sequence>
          </xs:complexType>
       </xs:element>
       <xs:element name="SetTrackConfigurationResponse">
          <xs:complexType>
             <xs:sequence/>
          </xs:complexType>
       </xs:element>
       <xs:element name="CreateRecordingJob">
          <xs:complexType>
             <xs:sequence>
                <xs:element name="JobConfiguration"
type="tt:RecordingJobConfiguration"/>
             </xs:sequence>
          </xs:complexType>
       </xs:element>
       <xs:element name="CreateRecordingJobResponse">
          <xs:complexType>
             <xs:sequence>
                <xs:element name="JobToken" type="tt:RecordingJobReference"/>
                <xs:element name="JobConfiguration"
type="tt:RecordingJobConfiguration"/>
             </xs:sequence>
          </xs:complexType>
       </xs:element>
       <xs:element name="DeleteRecordingJob">
          <xs:complexType>
             <xs:sequence>
                <xs:element name="JobToken" type="tt:RecordingJobReference"/>
             </xs:sequence>
          </xs:complexType>
       </xs:element>
       <xs:element name="DeleteRecordingJobResponse">
          <xs:complexType>
             <xs:sequence/>
          </xs:complexType>
       </xs:element>
       <xs:element name="GetRecordingJobs">
          <xs:complexType>
             <xs:sequence/>
          </xs:complexType>
       </xs:element>
       <xs:element name="GetRecordingJobsResponse">
          <xs:complexType>
             <xs:sequence>
                <xs:element name="JobItem" type="tt:GetRecordingJobsResponseItem"
minOccurs="0" maxOccurs="unbounded"/>
             </xs:sequence>
          </xs:complexType>
       </xs:element>
       <xs:element name="SetRecordingJobConfiguration">
          <xs:complexType>
             <xs:sequence>
                <xs:element name="JobToken" type="tt:RecordingJobReference"/>
                <xs:element name="JobConfiguration"
type="tt:RecordingJobConfiguration"/>
             </xs:sequence>
          </xs:complexType>
       </xs:element>
       <xs:element name="SetRecordingJobConfigurationResponse">
```

```
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="JobConfiguration"
type="tt:RecordingJobConfiguration"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetRecordingJobConfiguration">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="JobToken" type="tt:RecordingJobReference"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetRecordingJobConfigurationResponse">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="JobConfiguration"
type="tt:RecordingJobConfiguration"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="SetRecordingJobMode">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="JobToken" type="tt:RecordingJobReference"/>
                  <xs:element name="Mode" type="tt:RecordingJobMode"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="SetRecordingJobModeResponse">
            <xs:complexType>
               <xs:sequence/>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetRecordingJobState">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="JobToken" type="tt:RecordingJobReference"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetRecordingJobStateResponse">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="State" type="tt:RecordingJobStateInformation"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetRecordingOptions">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="RecordingToken" type="tt:RecordingReference"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetRecordingOptionsResponse">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="Options" type="trc:RecordingOptions"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="ExportRecordedData">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="StartPoint" type="xs:dateTime" minOccurs="0"/>
                  <xs:element name="EndPoint" type="xs:dateTime" minOccurs="0" />
                  <xs:element name="SearchScope" type="tt:SearchScope"/>
                  <xs:element name="FileFormat" type="xs:string"/>
                  <xs:element name="StorageDestination"
type="tt:StorageReferencePath"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="ExportRecordedDataResponse">
            <xs:complexType>
               <xs:sequence>
```

```
                  <xs:element name="OperationToken" type="tt:ReferenceToken"/>
                  <xs:element name="FileNames" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
                  <xs:element name="Extension" minOccurs="0">
                     <xs:complexType>
                        <xs:sequence>
                           <xs:any namespace="##any" processContents="lax"/>
                        </xs:sequence>
                     </xs:complexType>
                  </xs:element>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="StopExportRecordedData">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="OperationToken" type="tt:ReferenceToken"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="StopExportRecordedDataResponse">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="Progress" type="xs:float"/>
                  <xs:element name="FileProgressStatus" type="tt:ArrayOfFileProgress"/>
                  <xs:any namespace="##any" processContents="lax"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetExportRecordedDataState">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="OperationToken" type="tt:ReferenceToken"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetExportRecordedDataStateResponse">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="Progress" type="xs:float"/>
                  <xs:element name="FileProgressStatus" type="tt:ArrayOfFileProgress"/>
                  <xs:any namespace="##any" processContents="lax"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:complexType name="RecordingOptions">
            <xs:sequence>
               <xs:element name="Job" type="trc:JobOptions"/>
               <xs:element name="Track" type="trc:TrackOptions"/>
               <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
         </xs:complexType>
         <xs:complexType name="JobOptions">
            <xs:attribute name="Spare" type="xs:int"/>
            <xs:attribute name="CompatibleSources" type="tt:StringAttrList"/>
            <xs:anyAttribute processContents="lax"/>
         </xs:complexType>
         <xs:complexType name="TrackOptions">
            <xs:attribute name="SpareTotal" type="xs:int"/>
            <xs:attribute name="SpareVideo" type="xs:int"/>
            <xs:attribute name="SpareAudio" type="xs:int"/>
            <xs:attribute name="SpareMetadata" type="xs:int"/>
            <xs:anyAttribute processContents="lax"/>
         </xs:complexType>
      </xs:schema>
   </wsdl:types>
   <wsdl:message name="GetServiceCapabilitiesRequest">
      <wsdl:part name="parameters" element="trc:GetServiceCapabilities"/>
   </wsdl:message>
   <wsdl:message name="GetServiceCapabilitiesResponse">
      <wsdl:part name="parameters" element="trc:GetServiceCapabilitiesResponse"/>
   </wsdl:message>
   <wsdl:message name="CreateRecordingRequest">
      <wsdl:part name="parameters" element="trc:CreateRecording"/>
   </wsdl:message>
   <wsdl:message name="CreateRecordingResponse">
      <wsdl:part name="parameters" element="trc:CreateRecordingResponse"/>
```

```
</wsdl:message>
<wsdl:message name="DeleteRecordingRequest">
   <wsdl:part name="parameters" element="trc:DeleteRecording"/>
</wsdl:message>
<wsdl:message name="DeleteRecordingResponse">
   <wsdl:part name="parameters" element="trc:DeleteRecordingResponse"/>
</wsdl:message>
<wsdl:message name="GetRecordingsRequest">
   <wsdl:part name="parameters" element="trc:GetRecordings"/>
</wsdl:message>
<wsdl:message name="GetRecordingsResponse">
   <wsdl:part name="parameters" element="trc:GetRecordingsResponse"/>
</wsdl:message>
<wsdl:message name="SetRecordingConfigurationRequest">
   <wsdl:part name="parameters" element="trc:SetRecordingConfiguration"/>
</wsdl:message>
<wsdl:message name="SetRecordingConfigurationResponse">
   <wsdl:part name="parameters" element="trc:SetRecordingConfigurationResponse"/>
</wsdl:message>
<wsdl:message name="GetRecordingConfigurationRequest">
   <wsdl:part name="parameters" element="trc:GetRecordingConfiguration"/>
</wsdl:message>
<wsdl:message name="GetRecordingConfigurationResponse">
   <wsdl:part name="parameters" element="trc:GetRecordingConfigurationResponse"/>
</wsdl:message>
<wsdl:message name="GetRecordingOptionsRequest">
   <wsdl:part name="parameters" element="trc:GetRecordingOptions"/>
</wsdl:message>
<wsdl:message name="GetRecordingOptionsResponse">
   <wsdl:part name="parameters" element="trc:GetRecordingOptionsResponse"/>
</wsdl:message>
<wsdl:message name="CreateTrackRequest">
   <wsdl:part name="parameters" element="trc:CreateTrack"/>
</wsdl:message>
<wsdl:message name="CreateTrackResponse">
   <wsdl:part name="parameters" element="trc:CreateTrackResponse"/>
</wsdl:message>
<wsdl:message name="DeleteTrackRequest">
   <wsdl:part name="parameters" element="trc:DeleteTrack"/>
</wsdl:message>
<wsdl:message name="DeleteTrackResponse">
   <wsdl:part name="parameters" element="trc:DeleteTrackResponse"/>
</wsdl:message>
<wsdl:message name="GetTrackConfigurationRequest">
   <wsdl:part name="parameters" element="trc:GetTrackConfiguration"/>
</wsdl:message>
<wsdl:message name="GetTrackConfigurationResponse">
   <wsdl:part name="parameters" element="trc:GetTrackConfigurationResponse"/>
</wsdl:message>
<wsdl:message name="SetTrackConfigurationRequest">
   <wsdl:part name="parameters" element="trc:SetTrackConfiguration"/>
</wsdl:message>
<wsdl:message name="SetTrackConfigurationResponse">
   <wsdl:part name="parameters" element="trc:SetTrackConfigurationResponse"/>
</wsdl:message>
<wsdl:message name="CreateRecordingJobRequest">
   <wsdl:part name="parameters" element="trc:CreateRecordingJob"/>
</wsdl:message>
<wsdl:message name="CreateRecordingJobResponse">
   <wsdl:part name="parameters" element="trc:CreateRecordingJobResponse"/>
</wsdl:message>
<wsdl:message name="DeleteRecordingJobRequest">
   <wsdl:part name="parameters" element="trc:DeleteRecordingJob"/>
</wsdl:message>
<wsdl:message name="DeleteRecordingJobResponse">
   <wsdl:part name="parameters" element="trc:DeleteRecordingJobResponse"/>
</wsdl:message>
<wsdl:message name="GetRecordingJobsRequest">
   <wsdl:part name="parameters" element="trc:GetRecordingJobs"/>
</wsdl:message>
<wsdl:message name="GetRecordingJobsResponse">
   <wsdl:part name="parameters" element="trc:GetRecordingJobsResponse"/>
</wsdl:message>
<wsdl:message name="SetRecordingJobConfigurationRequest">
   <wsdl:part name="parameters" element="trc:SetRecordingJobConfiguration"/>
</wsdl:message>
<wsdl:message name="SetRecordingJobConfigurationResponse">
   <wsdl:part name="parameters" element="trc:SetRecordingJobConfigurationResponse"/>
```

```
</wsdl:message>
<wsdl:message name="GetRecordingJobConfigurationRequest">
   <wsdl:part name="parameters" element="trc:GetRecordingJobConfiguration"/>
</wsdl:message>
<wsdl:message name="GetRecordingJobConfigurationResponse">
   <wsdl:part name="parameters" element="trc:GetRecordingJobConfigurationResponse"/>
</wsdl:message>
<wsdl:message name="SetRecordingJobModeRequest">
   <wsdl:part name="parameters" element="trc:SetRecordingJobMode"/>
</wsdl:message>
<wsdl:message name="SetRecordingJobModeResponse">
   <wsdl:part name="parameters" element="trc:SetRecordingJobModeResponse"/>
</wsdl:message>
<wsdl:message name="GetRecordingJobStateRequest">
   <wsdl:part name="parameters" element="trc:GetRecordingJobState"/>
</wsdl:message>
<wsdl:message name="GetRecordingJobStateResponse">
   <wsdl:part name="parameters" element="trc:GetRecordingJobStateResponse"/>
</wsdl:message>
<wsdl:message name="ExportRecordedDataRequest">
   <wsdl:part name="parameters" element="trc:ExportRecordedData"/>
</wsdl:message>
<wsdl:message name="ExportRecordedDataResponse">
   <wsdl:part name="parameters" element="trc:ExportRecordedDataResponse"/>
</wsdl:message>
<wsdl:message name="StopExportRecordedDataRequest">
   <wsdl:part name="parameters" element="trc:StopExportRecordedData"/>
</wsdl:message>
<wsdl:message name="StopExportRecordedDataResponse">
   <wsdl:part name="parameters" element="trc:StopExportRecordedDataResponse"/>
</wsdl:message>
<wsdl:message name="GetExportRecordedDataStateRequest">
   <wsdl:part name="parameters" element="trc:GetExportRecordedDataState"/>
</wsdl:message>
<wsdl:message name="GetExportRecordedDataStateResponse">
   <wsdl:part name="parameters" element="trc:GetExportRecordedDataStateResponse"/>
</wsdl:message>
<wsdl:portType name="RecordingPort">
   <wsdl:operation name="GetServiceCapabilities">
      <wsdl:input message="trc:GetServiceCapabilitiesRequest"/>
      <wsdl:output message="trc:GetServiceCapabilitiesResponse"/>
   </wsdl:operation>
   <wsdl:operation name="CreateRecording">
      <wsdl:input message="trc:CreateRecordingRequest"/>
      <wsdl:output message="trc:CreateRecordingResponse"/>
   </wsdl:operation>
   <wsdl:operation name="DeleteRecording">
      <wsdl:input message="trc:DeleteRecordingRequest"/>
      <wsdl:output message="trc:DeleteRecordingResponse"/>
   </wsdl:operation>
   <wsdl:operation name="GetRecordings">
      <wsdl:input message="trc:GetRecordingsRequest"/>
      <wsdl:output message="trc:GetRecordingsResponse"/>
   </wsdl:operation>
   <wsdl:operation name="SetRecordingConfiguration">
      <wsdl:input message="trc:SetRecordingConfigurationRequest"/>
      <wsdl:output message="trc:SetRecordingConfigurationResponse"/>
   </wsdl:operation>
   <wsdl:operation name="GetRecordingConfiguration">
      <wsdl:input message="trc:GetRecordingConfigurationRequest"/>
      <wsdl:output message="trc:GetRecordingConfigurationResponse"/>
   </wsdl:operation>
   <wsdl:operation name="GetRecordingOptions">
      <wsdl:input message="trc:GetRecordingOptionsRequest"/>
      <wsdl:output message="trc:GetRecordingOptionsResponse"/>
   </wsdl:operation>
   <wsdl:operation name="CreateTrack">
      <wsdl:input message="trc:CreateTrackRequest"/>
      <wsdl:output message="trc:CreateTrackResponse"/>
   </wsdl:operation>
   <wsdl:operation name="DeleteTrack">
      <wsdl:input message="trc:DeleteTrackRequest"/>
      <wsdl:output message="trc:DeleteTrackResponse"/>
   </wsdl:operation>
   <wsdl:operation name="GetTrackConfiguration">
      <wsdl:input message="trc:GetTrackConfigurationRequest"/>
      <wsdl:output message="trc:GetTrackConfigurationResponse"/>
   </wsdl:operation>
```

```
      <wsdl:operation name="SetTrackConfiguration">
         <wsdl:input message="trc:SetTrackConfigurationRequest"/>
         <wsdl:output message="trc:SetTrackConfigurationResponse"/>
      </wsdl:operation>
      <wsdl:operation name="CreateRecordingJob">
         <wsdl:input message="trc:CreateRecordingJobRequest"/>
         <wsdl:output message="trc:CreateRecordingJobResponse"/>
      </wsdl:operation>
      <wsdl:operation name="DeleteRecordingJob">
         <wsdl:input message="trc:DeleteRecordingJobRequest"/>
         <wsdl:output message="trc:DeleteRecordingJobResponse"/>
      </wsdl:operation>
      <wsdl:operation name="GetRecordingJobs">
         <wsdl:input message="trc:GetRecordingJobsRequest"/>
         <wsdl:output message="trc:GetRecordingJobsResponse"/>
      </wsdl:operation>
      <wsdl:operation name="SetRecordingJobConfiguration">
         <wsdl:input message="trc:SetRecordingJobConfigurationRequest"/>
         <wsdl:output message="trc:SetRecordingJobConfigurationResponse"/>
      </wsdl:operation>
      <wsdl:operation name="GetRecordingJobConfiguration">
         <wsdl:input message="trc:GetRecordingJobConfigurationRequest"/>
         <wsdl:output message="trc:GetRecordingJobConfigurationResponse"/>
      </wsdl:operation>
      <wsdl:operation name="SetRecordingJobMode">
         <wsdl:input message="trc:SetRecordingJobModeRequest"/>
         <wsdl:output message="trc:SetRecordingJobModeResponse"/>
      </wsdl:operation>
      <wsdl:operation name="GetRecordingJobState">
         <wsdl:input message="trc:GetRecordingJobStateRequest"/>
         <wsdl:output message="trc:GetRecordingJobStateResponse"/>
      </wsdl:operation>
      <wsdl:operation name="ExportRecordedData">
         <wsdl:input message="trc:ExportRecordedDataRequest"/>
         <wsdl:output message="trc:ExportRecordedDataResponse"/>
      </wsdl:operation>
      <wsdl:operation name="StopExportRecordedData">
         <wsdl:input message="trc:StopExportRecordedDataRequest"/>
         <wsdl:output message="trc:StopExportRecordedDataResponse"/>
      </wsdl:operation>
      <wsdl:operation name="GetExportRecordedDataState">
         <wsdl:input message="trc:GetExportRecordedDataStateRequest"/>
         <wsdl:output message="trc:GetExportRecordedDataStateResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="RecordingBinding" type="trc:RecordingPort">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="GetServiceCapabilities">
         <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/GetServiceCapabilities"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="CreateRecording">
         <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/CreateRecording"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="DeleteRecording">
         <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/DeleteRecording"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetRecordings">
```

```
            <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/GetRecordings"/>
            <wsdl:input>
               <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
               <soap:body use="literal"/>
            </wsdl:output>
         </wsdl:operation>
         <wsdl:operation name="SetRecordingConfiguration">
            <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/SetRecordingConfiguration"/>
            <wsdl:input>
               <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
               <soap:body use="literal"/>
            </wsdl:output>
         </wsdl:operation>
         <wsdl:operation name="GetRecordingConfiguration">
            <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/GetRecordingConfiguration"/>
            <wsdl:input>
               <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
               <soap:body use="literal"/>
            </wsdl:output>
         </wsdl:operation>
         <wsdl:operation name="GetRecordingOptions">
            <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/GetRecordingOptions"/>
            <wsdl:input>
               <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
               <soap:body use="literal"/>
            </wsdl:output>
         </wsdl:operation>
         <wsdl:operation name="CreateTrack">
            <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/CreateTrack"/>
            <wsdl:input>
               <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
               <soap:body use="literal"/>
            </wsdl:output>
         </wsdl:operation>
         <wsdl:operation name="DeleteTrack">
            <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/DeleteTrack"/>
            <wsdl:input>
               <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
               <soap:body use="literal"/>
            </wsdl:output>
         </wsdl:operation>
         <wsdl:operation name="GetTrackConfiguration">
            <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/GetTrackConfiguration"/>
            <wsdl:input>
               <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
               <soap:body use="literal"/>
            </wsdl:output>
         </wsdl:operation>
         <wsdl:operation name="SetTrackConfiguration">
            <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/SetTrackConfiguration"/>
            <wsdl:input>
               <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
               <soap:body use="literal"/>
            </wsdl:output>
```

```
      </wsdl:operation>
      <wsdl:operation name="CreateRecordingJob">
         <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/CreateRecordingJob"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="DeleteRecordingJob">
         <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/DeleteRecordingJob"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetRecordingJobs">
         <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/GetRecordingJobs"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="SetRecordingJobConfiguration">
         <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/SetRecordingJobConfiguration"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetRecordingJobConfiguration">
         <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/GetRecordingJobConfiguration"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="SetRecordingJobMode">
         <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/SetRecordingJobMode"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetRecordingJobState">
         <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/GetRecordingJobState"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="ExportRecordedData">
         <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/ExportRecordedData"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
```

```
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="StopExportRecordedData">
         <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/StopExportRecordedData"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetExportRecordedDataState">
         <soap:operation
soapAction="http://www.onvif.org/ver10/recording/wsdl/GetExportRecordedDataState"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
</wsdl:definitions>
```

B.2 Search

```
<?xml version="1.0" encoding="utf-8"?><?xml-stylesheet type="text/xsl"
href="../ver20/util/onvif-wsdl-viewer.xsl"?><wsdl:definitions
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tse="http://www.onvif.org/ver10/search/wsdl"
targetNamespace="http://www.onvif.org/ver10/search/wsdl">
   <wsdl:types>
      <xs:schema xmlns:tt="http://www.onvif.org/ver10/schema"
targetNamespace="http://www.onvif.org/ver10/search/wsdl" elementFormDefault="qualified"
version="2.4.2">
         <xs:import namespace="http://www.onvif.org/ver10/schema"
schemaLocation="../ver10/schema/onvif.xsd"/>
         <xs:element name="GetServiceCapabilities">
            <xs:complexType>
               <xs:sequence/>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetServiceCapabilitiesResponse">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="Capabilities" type="tse:Capabilities"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:complexType name="Capabilities">
            <xs:sequence>
               <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="MetadataSearch" type="xs:boolean"/>
            <xs:attribute name="GeneralStartEvents" type="xs:boolean"/>
            <xs:anyAttribute processContents="lax"/>
         </xs:complexType>
         <xs:element name="Capabilities" type="tse:Capabilities"/>
         <xs:element name="GetRecordingSummary">
            <xs:complexType>
               <xs:sequence/>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetRecordingSummaryResponse">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="Summary" type="tt:RecordingSummary"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
```

```
            <xs:element name="GetRecordingInformation">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="RecordingToken" type="tt:RecordingReference"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="GetRecordingInformationResponse">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="RecordingInformation"
type="tt:RecordingInformation"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="GetMediaAttributes">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="RecordingTokens" type="tt:RecordingReference"
minOccurs="0" maxOccurs="unbounded"/>
                     <xs:element name="Time" type="xs:dateTime"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="GetMediaAttributesResponse">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="MediaAttributes" type="tt:MediaAttributes"
minOccurs="0" maxOccurs="unbounded"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="FindRecordings">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="Scope" type="tt:SearchScope"/>
                     <xs:element name="MaxMatches" type="xs:int" minOccurs="0"/>
                     <xs:element name="KeepAliveTime" type="xs:duration"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="FindRecordingsResponse">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="SearchToken" type="tt:JobToken"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="GetRecordingSearchResults">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="SearchToken" type="tt:JobToken"/>
                     <xs:element name="MinResults" type="xs:int" minOccurs="0"/>
                     <xs:element name="MaxResults" type="xs:int" minOccurs="0"/>
                     <xs:element name="WaitTime" type="xs:duration" minOccurs="0"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="GetRecordingSearchResultsResponse">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="ResultList" type="tt:FindRecordingResultList"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="FindEvents">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="StartPoint" type="xs:dateTime"/>
                     <xs:element name="EndPoint" type="xs:dateTime" minOccurs="0"/>
                     <xs:element name="Scope" type="tt:SearchScope"/>
                     <xs:element name="SearchFilter" type="tt:EventFilter"/>
                     <xs:element name="IncludeStartState" type="xs:boolean"/>
                     <xs:element name="MaxMatches" type="xs:int" minOccurs="0"/>
                     <xs:element name="KeepAliveTime" type="xs:duration"/>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
```

```
<xs:element name="FindEventsResponse">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="SearchToken" type="tt:JobToken"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="GetEventSearchResults">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="SearchToken" type="tt:JobToken"/>
         <xs:element name="MinResults" type="xs:int" minOccurs="0"/>
         <xs:element name="MaxResults" type="xs:int" minOccurs="0"/>
         <xs:element name="WaitTime" type="xs:duration" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="GetEventSearchResultsResponse">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="ResultList" type="tt:FindEventResultList"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="FindPTZPosition">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="StartPoint" type="xs:dateTime"/>
         <xs:element name="EndPoint" type="xs:dateTime" minOccurs="0"/>
         <xs:element name="Scope" type="tt:SearchScope"/>
         <xs:element name="SearchFilter" type="tt:PTZPositionFilter"/>
         <xs:element name="MaxMatches" type="xs:int" minOccurs="0"/>
         <xs:element name="KeepAliveTime" type="xs:duration"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="FindPTZPositionResponse">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="SearchToken" type="tt:JobToken"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="GetPTZPositionSearchResults">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="SearchToken" type="tt:JobToken"/>
         <xs:element name="MinResults" type="xs:int" minOccurs="0"/>
         <xs:element name="MaxResults" type="xs:int" minOccurs="0"/>
         <xs:element name="WaitTime" type="xs:duration" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="GetPTZPositionSearchResultsResponse">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="ResultList" type="tt:FindPTZPositionResultList"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="FindMetadata">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="StartPoint" type="xs:dateTime"/>
         <xs:element name="EndPoint" type="xs:dateTime" minOccurs="0"/>
         <xs:element name="Scope" type="tt:SearchScope"/>
         <xs:element name="MetadataFilter" type="tt:MetadataFilter"/>
         <xs:element name="MaxMatches" type="xs:int" minOccurs="0"/>
         <xs:element name="KeepAliveTime" type="xs:duration"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
<xs:element name="FindMetadataResponse">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="SearchToken" type="tt:JobToken"/>
      </xs:sequence>
   </xs:complexType>
```

```
      </xs:element>
      <xs:element name="GetMetadataSearchResults">
         <xs:complexType>
            <xs:sequence>
               <xs:element name="SearchToken" type="tt:JobToken"/>
               <xs:element name="MinResults" type="xs:int" minOccurs="0"/>
               <xs:element name="MaxResults" type="xs:int" minOccurs="0"/>
               <xs:element name="WaitTime" type="xs:duration" minOccurs="0"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
      <xs:element name="GetMetadataSearchResultsResponse">
         <xs:complexType>
            <xs:sequence>
               <xs:element name="ResultList" type="tt:FindMetadataResultList"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
      <xs:element name="GetSearchState">
         <xs:complexType>
            <xs:sequence>
               <xs:element name="SearchToken" type="tt:JobToken"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
      <xs:element name="GetSearchStateResponse">
         <xs:complexType>
            <xs:sequence>
               <xs:element name="State" type="tt:SearchState"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
      <xs:element name="EndSearch">
         <xs:complexType>
            <xs:sequence>
               <xs:element name="SearchToken" type="tt:JobToken"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
      <xs:element name="EndSearchResponse">
         <xs:complexType>
            <xs:sequence>
               <xs:element name="Endpoint" type="xs:dateTime"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
   </xs:schema>
</wsdl:types>
<wsdl:message name="GetServiceCapabilitiesRequest">
   <wsdl:part name="parameters" element="tse:GetServiceCapabilities"/>
</wsdl:message>
<wsdl:message name="GetServiceCapabilitiesResponse">
   <wsdl:part name="parameters" element="tse:GetServiceCapabilitiesResponse"/>
</wsdl:message>
<wsdl:message name="FindEventsRequest">
   <wsdl:part name="parameters" element="tse:FindEvents"/>
</wsdl:message>
<wsdl:message name="FindEventsResponse">
   <wsdl:part name="parameters" element="tse:FindEventsResponse"/>
</wsdl:message>
<wsdl:message name="GetEventSearchResultsRequest">
   <wsdl:part name="parameters" element="tse:GetEventSearchResults"/>
</wsdl:message>
<wsdl:message name="GetEventSearchResultsResponse">
   <wsdl:part name="parameters" element="tse:GetEventSearchResultsResponse"/>
</wsdl:message>
<wsdl:message name="GetSearchStateRequest">
   <wsdl:part name="parameters" element="tse:GetSearchState"/>
</wsdl:message>
<wsdl:message name="GetSearchStateResponse">
   <wsdl:part name="parameters" element="tse:GetSearchStateResponse"/>
</wsdl:message>
<wsdl:message name="EndSearchRequest">
   <wsdl:part name="parameters" element="tse:EndSearch"/>
</wsdl:message>
<wsdl:message name="EndSearchResponse">
   <wsdl:part name="parameters" element="tse:EndSearchResponse"/>
</wsdl:message>
```

```
<wsdl:message name="FindPTZPositionRequest">
   <wsdl:part name="parameters" element="tse:FindPTZPosition"/>
</wsdl:message>
<wsdl:message name="FindPTZPositionResponse">
   <wsdl:part name="parameters" element="tse:FindPTZPositionResponse"/>
</wsdl:message>
<wsdl:message name="GetPTZPositionSearchResultsRequest">
   <wsdl:part name="parameters" element="tse:GetPTZPositionSearchResults"/>
</wsdl:message>
<wsdl:message name="GetPTZPositionSearchResultsResponse">
   <wsdl:part name="parameters" element="tse:GetPTZPositionSearchResultsResponse"/>
</wsdl:message>
<wsdl:message name="GetRecordingSummaryRequest">
   <wsdl:part name="parameters" element="tse:GetRecordingSummary"/>
</wsdl:message>
<wsdl:message name="GetRecordingSummaryResponse">
   <wsdl:part name="parameters" element="tse:GetRecordingSummaryResponse"/>
</wsdl:message>
<wsdl:message name="GetRecordingInformationRequest">
   <wsdl:part name="parameters" element="tse:GetRecordingInformation"/>
</wsdl:message>
<wsdl:message name="GetRecordingInformationResponse">
   <wsdl:part name="parameters" element="tse:GetRecordingInformationResponse"/>
</wsdl:message>
<wsdl:message name="GetMediaAttributesRequest">
   <wsdl:part name="parameters" element="tse:GetMediaAttributes"/>
</wsdl:message>
<wsdl:message name="GetMediaAttributesResponse">
   <wsdl:part name="parameters" element="tse:GetMediaAttributesResponse"/>
</wsdl:message>
<wsdl:message name="FindRecordingsRequest">
   <wsdl:part name="parameters" element="tse:FindRecordings"/>
</wsdl:message>
<wsdl:message name="FindRecordingsResponse">
   <wsdl:part name="parameters" element="tse:FindRecordingsResponse"/>
</wsdl:message>
<wsdl:message name="GetRecordingSearchResultsRequest">
   <wsdl:part name="parameters" element="tse:GetRecordingSearchResults"/>
</wsdl:message>
<wsdl:message name="GetRecordingSearchResultsResponse">
   <wsdl:part name="parameters" element="tse:GetRecordingSearchResultsResponse"/>
</wsdl:message>
<wsdl:message name="FindMetadataRequest">
   <wsdl:part name="parameters" element="tse:FindMetadata"/>
</wsdl:message>
<wsdl:message name="FindMetadataResponse">
   <wsdl:part name="parameters" element="tse:FindMetadataResponse"/>
</wsdl:message>
<wsdl:message name="GetMetadataSearchResultsRequest">
   <wsdl:part name="parameters" element="tse:GetMetadataSearchResults"/>
</wsdl:message>
<wsdl:message name="GetMetadataSearchResultsResponse">
   <wsdl:part name="parameters" element="tse:GetMetadataSearchResultsResponse"/>
</wsdl:message>
<wsdl:portType name="SearchPort">
   <wsdl:operation name="GetServiceCapabilities">
      <wsdl:input message="tse:GetServiceCapabilitiesRequest"/>
      <wsdl:output message="tse:GetServiceCapabilitiesResponse"/>
   </wsdl:operation>
   <wsdl:operation name="GetRecordingSummary">
      <wsdl:input message="tse:GetRecordingSummaryRequest"/>
      <wsdl:output message="tse:GetRecordingSummaryResponse"/>
   </wsdl:operation>
   <wsdl:operation name="GetRecordingInformation">
      <wsdl:input message="tse:GetRecordingInformationRequest"/>
      <wsdl:output message="tse:GetRecordingInformationResponse"/>
   </wsdl:operation>
   <wsdl:operation name="GetMediaAttributes">
      <wsdl:input message="tse:GetMediaAttributesRequest"/>
      <wsdl:output message="tse:GetMediaAttributesResponse"/>
   </wsdl:operation>
   <wsdl:operation name="FindRecordings">
      <wsdl:input message="tse:FindRecordingsRequest"/>
      <wsdl:output message="tse:FindRecordingsResponse"/>
   </wsdl:operation>
   <wsdl:operation name="GetRecordingSearchResults">
      <wsdl:input message="tse:GetRecordingSearchResultsRequest"/>
      <wsdl:output message="tse:GetRecordingSearchResultsResponse"/>
```

```
      </wsdl:operation>
      <wsdl:operation name="FindEvents">
         <wsdl:input message="tse:FindEventsRequest"/>
         <wsdl:output message="tse:FindEventsResponse"/>
      </wsdl:operation>
      <wsdl:operation name="GetEventSearchResults">
         <wsdl:input message="tse:GetEventSearchResultsRequest"/>
         <wsdl:output message="tse:GetEventSearchResultsResponse"/>
      </wsdl:operation>
      <wsdl:operation name="FindPTZPosition">
         <wsdl:input message="tse:FindPTZPositionRequest"/>
         <wsdl:output message="tse:FindPTZPositionResponse"/>
      </wsdl:operation>
      <wsdl:operation name="GetPTZPositionSearchResults">
         <wsdl:input message="tse:GetPTZPositionSearchResultsRequest"/>
         <wsdl:output message="tse:GetPTZPositionSearchResultsResponse"/>
      </wsdl:operation>
      <wsdl:operation name="GetSearchState">
         <wsdl:input message="tse:GetSearchStateRequest"/>
         <wsdl:output message="tse:GetSearchStateResponse"/>
      </wsdl:operation>
      <wsdl:operation name="EndSearch">
         <wsdl:input message="tse:EndSearchRequest"/>
         <wsdl:output message="tse:EndSearchResponse"/>
      </wsdl:operation>
      <wsdl:operation name="FindMetadata">
         <wsdl:input message="tse:FindMetadataRequest"/>
         <wsdl:output message="tse:FindMetadataResponse"/>
      </wsdl:operation>
      <wsdl:operation name="GetMetadataSearchResults">
         <wsdl:input message="tse:GetMetadataSearchResultsRequest"/>
         <wsdl:output message="tse:GetMetadataSearchResultsResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="SearchBinding" type="tse:SearchPort">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="GetServiceCapabilities">
         <soap:operation
soapAction="http://www.onvif.org/ver10/search/wsdl/GetServiceCapabilities"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetRecordingSummary">
         <soap:operation
soapAction="http://www.onvif.org/ver10/search/wsdl/GetRecordingSummary"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetRecordingInformation">
         <soap:operation
soapAction="http://www.onvif.org/ver10/search/wsdl/GetRecordingInformation"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetMediaAttributes">
         <soap:operation
soapAction="http://www.onvif.org/ver10/search/wsdl/GetMediaAttributes"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="FindRecordings">
         <soap:operation
```

```
soapAction="http://www.onvif.org/ver10/search/wsdl/FindRecordings"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetRecordingSearchResults">
         <soap:operation
soapAction="http://www.onvif.org/ver10/search/wsdl/GetRecordingSearchResults"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="FindEvents">
         <soap:operation
soapAction="http://www.onvif.org/ver10/search/wsdl/FindEvents"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetEventSearchResults">
         <soap:operation
soapAction="http://www.onvif.org/ver10/search/wsdl/GetEventSearchResults"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="FindPTZPosition">
         <soap:operation
soapAction="http://www.onvif.org/ver10/search/wsdl/FindPTZPosition"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetPTZPositionSearchResults">
         <soap:operation
soapAction="http://www.onvif.org/ver10/search/wsdl/GetPTZPositionSearchResults"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetSearchState">
         <soap:operation
soapAction="http://www.onvif.org/ver10/search/wsdl/GetSearchState"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="EndSearch">
         <soap:operation
soapAction="http://www.onvif.org/ver10/search/wsdl/EndSearch"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
```

```
      <wsdl:operation name="FindMetadata">
         <soap:operation
soapAction="http://www.onvif.org/ver10/search/wsdl/FindMetadata"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetMetadataSearchResults">
         <soap:operation
soapAction="http://www.onvif.org/ver10/search/wsdl/GetMetadataSearchResults"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
</wsdl:definitions>
```

B.3 Replay control

```
<?xml version="1.0" encoding="utf-8"?><?xml-stylesheet type="text/xsl"
href="../ver20/util/onvif-wsdl-viewer.xsl"?><wsdl:definitions
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:trp="http://www.onvif.org/ver10/replay/wsdl"
targetNamespace="http://www.onvif.org/ver10/replay/wsdl">
   <wsdl:types>
      <xs:schema xmlns:tt="http://www.onvif.org/ver10/schema"
targetNamespace="http://www.onvif.org/ver10/replay/wsdl" elementFormDefault="qualified"
version="17.06">
         <xs:import namespace="http://www.onvif.org/ver10/schema"
schemaLocation="../ver10/schema/onvif.xsd"/>
         <xs:element name="GetServiceCapabilities">
            <xs:complexType>
               <xs:sequence/>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetServiceCapabilitiesResponse">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="Capabilities" type="trp:Capabilities"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:complexType name="Capabilities">
            <xs:sequence>
               <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="ReversePlayback" type="xs:boolean"/>
            <xs:attribute name="SessionTimeoutRange" type="tt:FloatAttrList"/>
            <xs:attribute name="RTP_RTSP_TCP" type="xs:boolean"/>
            <xs:attribute name="RTSPWebSocketUri" type="xs:anyURI"/>
            <xs:anyAttribute processContents="lax"/>
         </xs:complexType>
         <xs:element name="Capabilities" type="trp:Capabilities"/>
         <xs:element name="GetReplayUri">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="StreamSetup" type="tt:StreamSetup"/>
                  <xs:element name="RecordingToken" type="tt:ReferenceToken"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetReplayUriResponse">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="Uri" type="xs:anyURI"/>
               </xs:sequence>
```

```
            </xs:complexType>
         </xs:element>
         <xs:element name="SetReplayConfiguration">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="Configuration" type="tt:ReplayConfiguration"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="SetReplayConfigurationResponse">
            <xs:complexType>
               <xs:sequence/>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetReplayConfiguration">
            <xs:complexType>
               <xs:sequence/>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetReplayConfigurationResponse">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="Configuration" type="tt:ReplayConfiguration"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:schema>
   </wsdl:types>
   <wsdl:message name="GetServiceCapabilitiesRequest">
      <wsdl:part name="parameters" element="trp:GetServiceCapabilities"/>
   </wsdl:message>
   <wsdl:message name="GetServiceCapabilitiesResponse">
      <wsdl:part name="parameters" element="trp:GetServiceCapabilitiesResponse"/>
   </wsdl:message>
   <wsdl:message name="GetReplayUriRequest">
      <wsdl:part name="parameters" element="trp:GetReplayUri"/>
   </wsdl:message>
   <wsdl:message name="GetReplayUriResponse">
      <wsdl:part name="parameters" element="trp:GetReplayUriResponse"/>
   </wsdl:message>
   <wsdl:message name="SetReplayConfigurationRequest">
      <wsdl:part name="parameters" element="trp:SetReplayConfiguration"/>
   </wsdl:message>
   <wsdl:message name="SetReplayConfigurationResponse">
      <wsdl:part name="parameters" element="trp:SetReplayConfigurationResponse"/>
   </wsdl:message>
   <wsdl:message name="GetReplayConfigurationRequest">
      <wsdl:part name="parameters" element="trp:GetReplayConfiguration"/>
   </wsdl:message>
   <wsdl:message name="GetReplayConfigurationResponse">
      <wsdl:part name="parameters" element="trp:GetReplayConfigurationResponse"/>
   </wsdl:message>
   <wsdl:portType name="ReplayPort">
      <wsdl:operation name="GetServiceCapabilities">
         <wsdl:input message="trp:GetServiceCapabilitiesRequest"/>
         <wsdl:output message="trp:GetServiceCapabilitiesResponse"/>
      </wsdl:operation>
      <wsdl:operation name="GetReplayUri">
         <wsdl:input message="trp:GetReplayUriRequest"/>
         <wsdl:output message="trp:GetReplayUriResponse"/>
      </wsdl:operation>
      <wsdl:operation name="GetReplayConfiguration">
         <wsdl:input message="trp:GetReplayConfigurationRequest"/>
         <wsdl:output message="trp:GetReplayConfigurationResponse"/>
      </wsdl:operation>
      <wsdl:operation name="SetReplayConfiguration">
         <wsdl:input message="trp:SetReplayConfigurationRequest"/>
         <wsdl:output message="trp:SetReplayConfigurationResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="ReplayBinding" type="trp:ReplayPort">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="GetServiceCapabilities">
         <soap:operation
soapAction="http://www.onvif.org/ver10/replay/wsdl/GetServiceCapabilities"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
```

```
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetReplayUri">
         <soap:operation
soapAction="http://www.onvif.org/ver10/replay/wsdl/GetReplayUri"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetReplayConfiguration">
         <soap:operation
soapAction="http://www.onvif.org/ver10/replay/wsdl/GetReplayConfiguration"/>
         <wsdl:input>
            <soap:body parts="parameters" use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body parts="parameters" use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="SetReplayConfiguration">
         <soap:operation
soapAction="http://www.onvif.org/ver10/replay/wsdl/SetReplayConfiguration"/>
         <wsdl:input>
            <soap:body parts="parameters" use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body parts="parameters" use="literal"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
</wsdl:definitions>
```

B.4 Receiver

```
<?xml version="1.0" encoding="utf-8"?><?xml-stylesheet type="text/xsl"
href="../ver20/util/onvif-wsdl-viewer.xsl"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:trv="http://www.onvif.org/ver10/receiver/wsdl"
targetNamespace="http://www.onvif.org/ver10/receiver/wsdl">
   <wsdl:types>
      <xs:schema xmlns:tt="http://www.onvif.org/ver10/schema"
targetNamespace="http://www.onvif.org/ver10/receiver/wsdl"
elementFormDefault="qualified" version="2.1.1">
         <xs:import namespace="http://www.onvif.org/ver10/schema"
schemaLocation="../ver10/schema/onvif.xsd"/>
         <xs:element name="GetServiceCapabilities">
            <xs:complexType>
               <xs:sequence/>
            </xs:complexType>
         </xs:element>
         <xs:element name="GetServiceCapabilitiesResponse">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="Capabilities" type="trv:Capabilities"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:complexType name="Capabilities">
            <xs:sequence>
               <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="RTP_Multicast" type="xs:boolean"/>
            <xs:attribute name="RTP_TCP" type="xs:boolean"/>
            <xs:attribute name="RTP_RTSP_TCP" type="xs:boolean"/>
            <xs:attribute name="SupportedReceivers" type="xs:int" use="required"/>
            <xs:attribute name="MaximumRTSPURILength" type="xs:int"/>
            <xs:anyAttribute processContents="lax"/>
```

```
        </xs:complexType>
        <xs:element name="Capabilities" type="trv:Capabilities"/>
        <xs:element name="GetReceivers">
           <xs:complexType>
              <xs:sequence/>
           </xs:complexType>
        </xs:element>
        <xs:element name="GetReceiversResponse">
           <xs:complexType>
              <xs:sequence>
                 <xs:element name="Receivers" type="tt:Receiver" minOccurs="0"
maxOccurs="unbounded"/>
              </xs:sequence>
           </xs:complexType>
        </xs:element>
        <xs:element name="GetReceiver">
           <xs:complexType>
              <xs:sequence>
                 <xs:element name="ReceiverToken" type="tt:ReferenceToken"/>
              </xs:sequence>
           </xs:complexType>
        </xs:element>
        <xs:element name="GetReceiverResponse">
           <xs:complexType>
              <xs:sequence>
                 <xs:element name="Receiver" type="tt:Receiver"/>
              </xs:sequence>
           </xs:complexType>
        </xs:element>
        <xs:element name="CreateReceiver">
           <xs:complexType>
              <xs:sequence>
                 <xs:element name="Configuration" type="tt:ReceiverConfiguration"/>
              </xs:sequence>
           </xs:complexType>
        </xs:element>
        <xs:element name="CreateReceiverResponse">
           <xs:complexType>
              <xs:sequence>
                 <xs:element name="Receiver" type="tt:Receiver"/>
              </xs:sequence>
           </xs:complexType>
        </xs:element>
        <xs:element name="DeleteReceiver">
           <xs:complexType>
              <xs:sequence>
                 <xs:element name="ReceiverToken" type="tt:ReferenceToken"/>
              </xs:sequence>
           </xs:complexType>
        </xs:element>
        <xs:element name="DeleteReceiverResponse">
           <xs:complexType>
              <xs:sequence/>
           </xs:complexType>
        </xs:element>
        <xs:element name="ConfigureReceiver">
           <xs:complexType>
              <xs:sequence>
                 <xs:element name="ReceiverToken" type="tt:ReferenceToken"/>
                 <xs:element name="Configuration" type="tt:ReceiverConfiguration"/>
              </xs:sequence>
           </xs:complexType>
        </xs:element>
        <xs:element name="ConfigureReceiverResponse">
           <xs:complexType>
              <xs:sequence/>
           </xs:complexType>
        </xs:element>
        <xs:element name="SetReceiverMode">
           <xs:complexType>
              <xs:sequence>
                 <xs:element name="ReceiverToken" type="tt:ReferenceToken"/>
                 <xs:element name="Mode" type="tt:ReceiverMode"/>
              </xs:sequence>
           </xs:complexType>
        </xs:element>
        <xs:element name="SetReceiverModeResponse">
           <xs:complexType>
```

```
            <xs:sequence/>
         </xs:complexType>
      </xs:element>
      <xs:element name="GetReceiverState">
         <xs:complexType>
            <xs:sequence>
               <xs:element name="ReceiverToken" type="tt:ReferenceToken"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
      <xs:element name="GetReceiverStateResponse">
         <xs:complexType>
            <xs:sequence>
               <xs:element name="ReceiverState" type="tt:ReceiverStateInformation"/>
            </xs:sequence>
         </xs:complexType>
      </xs:element>
   </xs:schema>
</wsdl:types>
<wsdl:message name="GetServiceCapabilitiesRequest">
   <wsdl:part name="parameters" element="trv:GetServiceCapabilities"/>
</wsdl:message>
<wsdl:message name="GetServiceCapabilitiesResponse">
   <wsdl:part name="parameters" element="trv:GetServiceCapabilitiesResponse"/>
</wsdl:message>
<wsdl:message name="GetReceiversRequest">
   <wsdl:part name="parameters" element="trv:GetReceivers"/>
</wsdl:message>
<wsdl:message name="GetReceiversResponse">
   <wsdl:part name="parameters" element="trv:GetReceiversResponse"/>
</wsdl:message>
<wsdl:message name="GetReceiverRequest">
   <wsdl:part name="parameters" element="trv:GetReceiver"/>
</wsdl:message>
<wsdl:message name="GetReceiverResponse">
   <wsdl:part name="parameters" element="trv:GetReceiverResponse"/>
</wsdl:message>
<wsdl:message name="CreateReceiverRequest">
   <wsdl:part name="parameters" element="trv:CreateReceiver"/>
</wsdl:message>
<wsdl:message name="CreateReceiverResponse">
   <wsdl:part name="parameters" element="trv:CreateReceiverResponse"/>
</wsdl:message>
<wsdl:message name="DeleteReceiverRequest">
   <wsdl:part name="parameters" element="trv:DeleteReceiver"/>
</wsdl:message>
<wsdl:message name="DeleteReceiverResponse">
   <wsdl:part name="parameters" element="trv:DeleteReceiverResponse"/>
</wsdl:message>
<wsdl:message name="ConfigureReceiverRequest">
   <wsdl:part name="parameters" element="trv:ConfigureReceiver"/>
</wsdl:message>
<wsdl:message name="ConfigureReceiverResponse">
   <wsdl:part name="parameters" element="trv:ConfigureReceiverResponse"/>
</wsdl:message>
<wsdl:message name="SetReceiverModeRequest">
   <wsdl:part name="parameters" element="trv:SetReceiverMode"/>
</wsdl:message>
<wsdl:message name="SetReceiverModeResponse">
   <wsdl:part name="parameters" element="trv:SetReceiverModeResponse"/>
</wsdl:message>
<wsdl:message name="GetReceiverStateRequest">
   <wsdl:part name="parameters" element="trv:GetReceiverState"/>
</wsdl:message>
<wsdl:message name="GetReceiverStateResponse">
   <wsdl:part name="parameters" element="trv:GetReceiverStateResponse"/>
</wsdl:message>
<wsdl:portType name="ReceiverPort">
   <wsdl:operation name="GetServiceCapabilities">
      <wsdl:input message="trv:GetServiceCapabilitiesRequest"/>
      <wsdl:output message="trv:GetServiceCapabilitiesResponse"/>
   </wsdl:operation>
   <wsdl:operation name="GetReceivers">
      <wsdl:input message="trv:GetReceiversRequest"/>
      <wsdl:output message="trv:GetReceiversResponse"/>
   </wsdl:operation>
   <wsdl:operation name="GetReceiver">
      <wsdl:input message="trv:GetReceiverRequest"/>
```

```
         <wsdl:output message="trv:GetReceiverResponse"/>
      </wsdl:operation>
      <wsdl:operation name="CreateReceiver">
         <wsdl:input message="trv:CreateReceiverRequest"/>
         <wsdl:output message="trv:CreateReceiverResponse"/>
      </wsdl:operation>
      <wsdl:operation name="DeleteReceiver">
         <wsdl:input message="trv:DeleteReceiverRequest"/>
         <wsdl:output message="trv:DeleteReceiverResponse"/>
      </wsdl:operation>
      <wsdl:operation name="ConfigureReceiver">
         <wsdl:input message="trv:ConfigureReceiverRequest"/>
         <wsdl:output message="trv:ConfigureReceiverResponse"/>
      </wsdl:operation>
      <wsdl:operation name="SetReceiverMode">
         <wsdl:input message="trv:SetReceiverModeRequest"/>
         <wsdl:output message="trv:SetReceiverModeResponse"/>
      </wsdl:operation>
      <wsdl:operation name="GetReceiverState">
         <wsdl:input message="trv:GetReceiverStateRequest"/>
         <wsdl:output message="trv:GetReceiverStateResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="ReceiverBinding" type="trv:ReceiverPort">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="GetServiceCapabilities">
         <soap:operation
soapAction="http://www.onvif.org/ver10/receiver/wsdl/GetServiceCapabilities"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetReceivers">
         <soap:operation
soapAction="http://www.onvif.org/ver10/receiver/wsdl/GetReceivers"/>
         <wsdl:input>
            <soap:body parts="parameters" use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body parts="parameters" use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="GetReceiver">
         <soap:operation
soapAction="http://www.onvif.org/ver10/receiver/wsdl/GetReceiver"/>
         <wsdl:input>
            <soap:body parts="parameters" use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body parts="parameters" use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="CreateReceiver">
         <soap:operation
soapAction="http://www.onvif.org/ver10/receiver/wsdl/CreateReceiver"/>
         <wsdl:input>
            <soap:body parts="parameters" use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body parts="parameters" use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="DeleteReceiver">
         <soap:operation
soapAction="http://www.onvif.org/ver10/receiver/wsdl/DeleteReceiver"/>
         <wsdl:input>
            <soap:body parts="parameters" use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body parts="parameters" use="literal"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="ConfigureReceiver">
         <soap:operation
soapAction="http://www.onvif.org/ver10/receiver/wsdl/ConfigureReceiver"/>
```

```
            <wsdl:input>
               <soap:body parts="parameters" use="literal"/>
            </wsdl:input>
            <wsdl:output>
               <soap:body parts="parameters" use="literal"/>
            </wsdl:output>
         </wsdl:operation>
         <wsdl:operation name="SetReceiverMode">
            <soap:operation
soapAction="http://www.onvif.org/ver10/receiver/wsdl/SetReceiverMode"/>
            <wsdl:input>
               <soap:body parts="parameters" use="literal"/>
            </wsdl:input>
            <wsdl:output>
               <soap:body parts="parameters" use="literal"/>
            </wsdl:output>
         </wsdl:operation>
         <wsdl:operation name="GetReceiverState">
            <soap:operation
soapAction="http://www.onvif.org/ver10/receiver/wsdl/GetReceiverState"/>
            <wsdl:input>
               <soap:body parts="parameters" use="literal"/>
            </wsdl:input>
            <wsdl:output>
               <soap:body parts="parameters" use="literal"/>
            </wsdl:output>
         </wsdl:operation>
      </wsdl:binding>
</wsdl:definitions>
```

B.5 Common Schema

This clause extends the generic data types as defined by Clause B.4 of IEC 60839-11-31:2016 and Clause F.7 of IEC 62676-2-31:2019.

```
<?xml version="1.0"?>
<xs:schema xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xmime="http://www.w3.org/2005/05/xmlmime"
xmlns:wsnt=http://docs.oasis-open.org/wsn/b-2
xmlns:xop="http://www.w3.org/2004/08/xop/include"
xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
targetNamespace="http://www.onvif.org/ver10/schema" elementFormDefault="qualified">
      <xs:complexType name="Receiver">
            <xs:sequence>
                  <xs:element name="Token" type="tt:ReferenceToken"/>
                  <xs:element name="Configuration" type="tt:ReceiverConfiguration"/>
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="ReceiverConfiguration">
            <xs:sequence>
                  <xs:element name="Mode" type="tt:ReceiverMode"/>
                  <xs:element name="MediaUri" type="xs:anyURI"/>
                  <xs:element name="StreamSetup" type="tt:StreamSetup"/>
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:simpleType name="ReceiverMode">
            <xs:restriction base="xs:string">
                  <xs:enumeration value="AutoConnect"/>
                  <xs:enumeration value="AlwaysConnect"/>
                  <xs:enumeration value="NeverConnect"/>
                  <xs:enumeration value="Unknown"/>
            </xs:restriction>
      </xs:simpleType>
      <xs:simpleType name="ReceiverState">
            <xs:restriction base="xs:string">
                  <xs:enumeration value="NotConnected"/>
                  <xs:enumeration value="Connecting"/>
                  <xs:enumeration value="Connected"/>
```

```
                <xs:enumeration value="Unknown"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:complexType name="ReceiverStateInformation">
            <xs:sequence>
                <xs:element name="State" type="tt:ReceiverState"/>
                <xs:element name="AutoCreated" type="xs:boolean"/>
                <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
        </xs:complexType>
        <xs:simpleType name="ReceiverReference">
            <xs:restriction base="tt:ReferenceToken"/>
        </xs:simpleType>
        <xs:simpleType name="RecordingReference">
            <xs:restriction base="tt:ReferenceToken"/>
        </xs:simpleType>
        <xs:complexType name="SourceReference">
            <xs:sequence>
                <xs:element name="Token" type="tt:ReferenceToken"/>
                <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="Type" type="xs:anyURI" use="optional"
default="http://www.onvif.org/ver10/schema/Receiver"/>
            <xs:anyAttribute processContents="lax"/>
        </xs:complexType>
        <xs:simpleType name="TrackReference">
            <xs:restriction base="tt:ReferenceToken"/>
        </xs:simpleType>
        <xs:simpleType name="Description">
            <xs:restriction base="xs:string"/>
        </xs:simpleType>
        <xs:complexType name="RecordingSummary">
            <xs:sequence>
                <xs:element name="DataFrom" type="xs:dateTime"/>
                <xs:element name="DataUntil" type="xs:dateTime"/>
                <xs:element name="NumberRecordings" type="xs:int"/>
                <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
        </xs:complexType>
        <xs:complexType name="SearchScope">
            <xs:sequence>
                <xs:element name="IncludedSources" type="tt:SourceReference" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="IncludedRecordings" type="tt:RecordingReference"
minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="RecordingInformationFilter" type="tt:XPathExpression"
minOccurs="0"/>
                <xs:element name="Extension" type="tt:SearchScopeExtension" minOccurs="0"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
        </xs:complexType>
        <xs:complexType name="SearchScopeExtension">
            <xs:sequence>
                <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="EventFilter">
            <xs:complexContent>
                <xs:extension base="wsnt:FilterType">
                    <xs:anyAttribute processContents="lax"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
        <xs:complexType name="PTZPositionFilter">
            <xs:sequence>
                <xs:element name="MinPosition" type="tt:PTZVector"/>
                <xs:element name="MaxPosition" type="tt:PTZVector"/>
                <xs:element name="EnterOrExit" type="xs:boolean"/>
                <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
```

```
      </xs:complexType>
      <xs:complexType name="MetadataFilter">
            <xs:sequence>
                  <xs:element name="MetadataStreamFilter" type="tt:XPathExpression"/>
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:simpleType name="XPathExpression">
            <xs:restriction base="xs:string"/>
      </xs:simpleType>
      <xs:complexType name="FindRecordingResultList">
            <xs:sequence>
                  <xs:element name="SearchState" type="tt:SearchState"/>
                  <xs:element name="RecordingInformation" type="tt:RecordingInformation"
minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="FindEventResultList">
            <xs:sequence>
                  <xs:element name="SearchState" type="tt:SearchState"/>
                  <xs:element name="Result" type="tt:FindEventResult" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="FindEventResult">
            <xs:sequence>
                  <xs:element name="RecordingToken" type="tt:RecordingReference"/>
                  <xs:element name="TrackToken" type="tt:TrackReference"/>
                  <xs:element name="Time" type="xs:dateTime"/>
                  <xs:element name="Event" type="wsnt:NotificationMessageHolderType"/>
                  <xs:element name="StartStateEvent" type="xs:boolean"/>
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="FindPTZPositionResultList">
            <xs:sequence>
                  <xs:element name="SearchState" type="tt:SearchState"/>
                  <xs:element name="Result" type="tt:FindPTZPositionResult" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="FindPTZPositionResult">
            <xs:sequence>
                  <xs:element name="RecordingToken" type="tt:RecordingReference"/>
                  <xs:element name="TrackToken" type="tt:TrackReference"/>
                  <xs:element name="Time" type="xs:dateTime"/>
                  <xs:element name="Position" type="tt:PTZVector"/>
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="FindMetadataResultList">
            <xs:sequence>
                  <xs:element name="SearchState" type="tt:SearchState"/>
                  <xs:element name="Result" type="tt:FindMetadataResult" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="FindMetadataResult">
            <xs:sequence>
                  <xs:element name="RecordingToken" type="tt:RecordingReference"/>
                  <xs:element name="TrackToken" type="tt:TrackReference"/>
                  <xs:element name="Time" type="xs:dateTime"/>
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:simpleType name="SearchState">
            <xs:restriction base="xs:string">
                  <xs:enumeration value="Queued"/>
                  <xs:enumeration value="Searching"/>
                  <xs:enumeration value="Completed"/>
```

```
                <xs:enumeration value="Unknown"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="JobToken">
            <xs:restriction base="tt:ReferenceToken"/>
        </xs:simpleType>
        <xs:complexType name="RecordingInformation">
            <xs:sequence>
                <xs:element name="RecordingToken" type="tt:RecordingReference"/>
                <xs:element name="Source" type="tt:RecordingSourceInformation"/>
                <xs:element name="EarliestRecording" type="xs:dateTime" minOccurs="0"/>
                <xs:element name="LatestRecording" type="xs:dateTime" minOccurs="0"/>
                <xs:element name="Content" type="tt:Description"/>
                <xs:element name="Track" type="tt:TrackInformation" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="RecordingStatus" type="tt:RecordingStatus"/>
                <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
        </xs:complexType>
        <xs:complexType name="RecordingSourceInformation">
            <xs:sequence>
                <xs:element name="SourceId" type="xs:anyURI"/>
                <xs:element name="Name" type="tt:Name"/>
                <xs:element name="Location" type="tt:Description"/>
                <xs:element name="Description" type="tt:Description"/>
                <xs:element name="Address" type="xs:anyURI"/>
                <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
        </xs:complexType>
        <xs:simpleType name="RecordingStatus">
            <xs:restriction base="xs:string">
                <xs:enumeration value="Initiated"/>
                <xs:enumeration value="Recording"/>
                <xs:enumeration value="Stopped"/>
                <xs:enumeration value="Removing"/>
                <xs:enumeration value="Removed"/>
                <xs:enumeration value="Unknown"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:complexType name="TrackInformation">
            <xs:sequence>
                <xs:element name="TrackToken" type="tt:TrackReference"/>
                <xs:element name="TrackType" type="tt:TrackType"/>
                <xs:element name="Description" type="tt:Description"/>
                <xs:element name="DataFrom" type="xs:dateTime"/>
                <xs:element name="DataTo" type="xs:dateTime"/>
                <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
        </xs:complexType>
        <xs:simpleType name="TrackType">
            <xs:restriction base="xs:string">
                <xs:enumeration value="Video"/>
                <xs:enumeration value="Audio"/>
                <xs:enumeration value="Metadata"/>
                <xs:enumeration value="Extended"/>
            </xs:restriction>
        </xs:simpleType>
        <xs:complexType name="MediaAttributes">
            <xs:sequence>
                <xs:element name="RecordingToken" type="tt:RecordingReference"/>
                <xs:element name="TrackAttributes" type="tt:TrackAttributes" minOccurs="0"
maxOccurs="unbounded"/>
                <xs:element name="From" type="xs:dateTime"/>
                <xs:element name="Until" type="xs:dateTime"/>
                <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
        </xs:complexType>
        <xs:complexType name="TrackAttributes">
            <xs:sequence>
                <xs:element name="TrackInformation" type="tt:TrackInformation"/>
```

```
                <xs:element name="VideoAttributes" type="tt:VideoAttributes" minOccurs="0"/>
                <xs:element name="AudioAttributes" type="tt:AudioAttributes" minOccurs="0"/>
                <xs:element name="MetadataAttributes" type="tt:MetadataAttributes"
minOccurs="0"/>
                <xs:element name="Extension" type="tt:TrackAttributesExtension"
minOccurs="0"/>
          </xs:sequence>
          <xs:anyAttribute processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="TrackAttributesExtension">
          <xs:sequence>
                <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
    </xs:complexType>
    <xs:complexType name="VideoAttributes">
          <xs:sequence>
                <xs:element name="Bitrate" type="xs:int" minOccurs="0"/>
                <xs:element name="Width" type="xs:int"/>
                <xs:element name="Height" type="xs:int"/>
                <xs:element name="Encoding" type="xs:string"/>
                <xs:element name="Framerate" type="xs:float"/>
                <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:anyAttribute processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="AudioAttributes">
          <xs:sequence>
                <xs:element name="Bitrate" type="xs:int" minOccurs="0"/>
                <xs:element name="Encoding" type="xs:string"/>
                <xs:element name="Samplerate" type="xs:int"/>
                <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:anyAttribute processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="MetadataAttributes">
          <xs:sequence>
                <xs:element name="CanContainPTZ" type="xs:boolean"/>
                <xs:element name="CanContainAnalytics" type="xs:boolean"/>
                <xs:element name="CanContainNotifications" type="xs:boolean"/>
                <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:attribute name="PtzSpaces" type="tt:StringAttrList"/>
          <xs:anyAttribute processContents="lax"/>
    </xs:complexType>
    <xs:simpleType name="RecordingJobReference">
          <xs:restriction base="tt:ReferenceToken"/>
    </xs:simpleType>
    <xs:complexType name="RecordingConfiguration">
          <xs:sequence>
                <xs:element name="Source" type="tt:RecordingSourceInformation"/>
                <xs:element name="Content" type="tt:Description"/>
                <xs:element name="MaximumRetentionTime" type="xs:duration"/>
                <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:anyAttribute processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="TrackConfiguration">
          <xs:sequence>
                <xs:element name="TrackType" type="tt:TrackType"/>
                <xs:element name="Description" type="tt:Description"/>
                <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
          <xs:anyAttribute processContents="lax"/>
    </xs:complexType>
    <xs:complexType name="GetRecordingsResponseItem">
          <xs:sequence>
                <xs:element name="RecordingToken" type="tt:RecordingReference"/>
                <xs:element name="Configuration" type="tt:RecordingConfiguration"/>
                <xs:element name="Tracks" type="tt:GetTracksResponseList"/>
                <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
```

```
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="GetTracksResponseList">
            <xs:sequence>
                  <xs:element name="Track" type="tt:GetTracksResponseItem" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="GetTracksResponseItem">
            <xs:sequence>
                  <xs:element name="TrackToken" type="tt:TrackReference"/>
                  <xs:element name="Configuration" type="tt:TrackConfiguration"/>
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="RecordingJobConfiguration">
            <xs:sequence>
                  <xs:element name="RecordingToken" type="tt:RecordingReference"/>
                  <xs:element name="Mode" type="tt:RecordingJobMode"/>
                  <xs:element name="Priority" type="xs:int"/>
                  <xs:element name="Source" type="tt:RecordingJobSource" minOccurs="0"
maxOccurs="unbounded"/>
                  <xs:element name="Extension" type="tt:RecordingJobConfigurationExtension"
minOccurs="0"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:simpleType name="RecordingJobMode">
            <xs:restriction base="xs:string"/>
      </xs:simpleType>
      <xs:complexType name="RecordingJobConfigurationExtension">
            <xs:sequence>
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="RecordingJobSource">
            <xs:sequence>
                  <xs:element name="SourceToken" type="tt:SourceReference" minOccurs="0"/>
                  <xs:element name="AutoCreateReceiver" type="xs:boolean" minOccurs="0"/>
                  <xs:element name="Tracks" type="tt:RecordingJobTrack" minOccurs="0"
maxOccurs="unbounded"/>
                  <xs:element name="Extension" type="tt:RecordingJobSourceExtension"
minOccurs="0"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="RecordingJobSourceExtension">
            <xs:sequence>
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
      <xs:complexType name="RecordingJobTrack">
            <xs:sequence>
                  <xs:element name="SourceTag" type="xs:string"/>
                  <xs:element name="Destination" type="tt:TrackReference"/>
                  <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="RecordingJobStateInformation">
            <xs:sequence>
                  <xs:element name="RecordingToken" type="tt:RecordingReference"/>
                  <xs:element name="State" type="tt:RecordingJobState"/>
                  <xs:element name="Sources" type="tt:RecordingJobStateSource" minOccurs="0"
maxOccurs="unbounded"/>
                  <xs:element name="Extension" type="tt:RecordingJobStateInformationExtension"
minOccurs="0"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="RecordingJobStateInformationExtension">
            <xs:sequence>
```

```
                <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
      </xs:complexType>
      <xs:simpleType name="RecordingJobState">
            <xs:restriction base="xs:string"/>
      </xs:simpleType>
      <xs:complexType name="RecordingJobStateSource">
            <xs:sequence>
                  <xs:element name="SourceToken" type="tt:SourceReference"/>
                  <xs:element name="State" type="tt:RecordingJobState"/>
                  <xs:element name="Tracks" type="tt:RecordingJobStateTracks"/>
                  <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="RecordingJobStateTracks">
            <xs:sequence>
                  <xs:element name="Track" type="tt:RecordingJobStateTrack" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="RecordingJobStateTrack">
            <xs:sequence>
                  <xs:element name="SourceTag" type="xs:string"/>
                  <xs:element name="Destination" type="tt:TrackReference"/>
                  <xs:element name="Error" type="xs:string" minOccurs="0"/>
                  <xs:element name="State" type="tt:RecordingJobState"/>
                  <xs:any namespace="##targetNamespace" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="GetRecordingJobsResponseItem">
            <xs:sequence>
                  <xs:element name="JobToken" type="tt:RecordingJobReference"/>
                  <xs:element name="JobConfiguration" type="tt:RecordingJobConfiguration"/>
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="ReplayConfiguration">
            <xs:sequence>
                  <xs:element name="SessionTimeout" type="xs:duration"/>
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="FileProgress">
            <xs:sequence>
                  <xs:element name="FileName" type="xs:string"/>
                  <xs:element name="Progress" type="xs:float"/>
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="ArrayOfFileProgress">
            <xs:sequence>
                  <xs:element name="FileProgress" type="tt:FileProgress" minOccurs="0"
maxOccurs="unbounded"/>
                  <xs:element name="Extension" type="tt:ArrayOfFileProgressExtension"
minOccurs="0"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="ArrayOfFileProgressExtension">
            <xs:sequence>
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="StorageReferencePath">
```

```
            <xs:sequence>
                  <xs:element name="StorageToken" type="tt:ReferenceToken"/>
                  <xs:element name="RelativePath" type="xs:string" minOccurs="0"/>
                  <xs:element name="Extension" type="tt:StorageReferencePathExtension"
minOccurs="0"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
      <xs:complexType name="StorageReferencePathExtension">
            <xs:sequence>
                  <xs:any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:anyAttribute processContents="lax"/>
      </xs:complexType>
</xs:schema>
```

Bibliography

ISO/IEC 14496-12, *Information technology – Coding of audio-visual objects – Part 12: ISO base media file format*

ISO/IEC 23000-10, *Information technology – Multimedia application format – Part 10: Surveillance application format*

FIPS 186-4, *Digital Signature Standard (DSS)*
<https://csrc.nist.gov/publications/detail/fips/186/4/final>

IETF RFC 1305, *Network Time Protocol (Version 3) – Specification, Implementation and Analysis*
<http://www.ietf.org/rfc/rfc1305.txt >

IETF RFC 4585, *Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)*
<http://www.ietf.org/rfc/rfc4585.txt >

SOMMAIRE

AVANT-PROPOS 116
INTRODUCTION 118
1 Domaine d'application 119
2 Références normatives 119
3 Termes, définitions et termes abrégés 120
3.1 Termes et définitions 120
3.2 Termes abrégés 121
4 Vue d'ensemble 121
4.1 Interfaces 121
4.2 Modèle de stockage 123
4.3 Contrôle d'enregistrement 124
4.4 Recherche 124
4.5 Contrôle de lecture 125
4.6 Format de fichiers d'exportation 125
4.6.1 Présentation 125
4.6.2 Cas d'utilisation 1: Restitution de vidéoclips morcelés et surdimensionnés à distance 126
4.6.3 Cas d'utilisation 2: Analyse criminalistique dans un tribunal 126
4.6.4 Cas d'utilisation 3: Restitution avec des lecteurs non équipés selon la présente spécification 127
4.7 Récepteur 127
5 Service de contrôle d'enregistrement 127
5.1 Vue d'ensemble 127
5.2 Exigences générales 129
5.3 Structures de données 129
5.3.1 RecordingConfiguration 129
5.3.2 TrackConfiguration 129
5.3.3 RecordingJobConfiguration 130
5.4 CreateRecording 131
5.5 DeleteRecording 132
5.6 GetRecordings 133
5.7 SetRecordingConfiguration 133
5.8 GetRecordingConfiguration 134
5.9 CreateTrack 134
5.10 DeleteTrack 135
5.11 GetTrackConfiguration 136
5.12 SetTrackConfiguration 136
5.13 CreateRecordingJob 137
5.14 DeleteRecordingJob 138
5.15 GetRecordingJobs 138
5.16 SetRecordingJobConfiguration 139
5.17 GetRecordingJobConfiguration 139
5.18 SetRecordingJobMode 140
5.19 GetRecordingJobState 140
5.20 GetRecordingOptions 142
5.21 ExportRecordedData 143
5.22 StopExportRecordedData 144

5.23 GetExportRecordedDataState 145
5.24 GetServiceCapabilities 145
5.25 Événements 147
5.25.1 Généralités 147
5.25.2 Modifications d'état des travaux d'enregistrement 147
5.25.3 Modifications de configuration 147
5.25.4 Suppression de données 148
5.25.5 Enregistrement et création et suppression de pistes 148
5.26 Exemples 149
5.26.1 Exemple 1: configuration de l'enregistrement d'une seule caméra 149
5.26.2 Exemple 2: enregistrement de plusieurs flux d'une caméra vers un seul enregistrement 149
6 Service de recherche 150
6.1 Généralités 150
6.2 Concepts 151
6.2.1 Direction de recherche 151
6.2.2 Événement d'enregistrement 151
6.2.3 Session de recherche 151
6.2.4 Étendue de la recherche 152
6.2.5 Filtres de recherche 152
6.2.6 Informations temporelles 152
6.3 Structures de données 152
6.3.1 Structure RecordingInformation 152
6.3.2 Structure RecordingSourceInformation 153
6.3.3 Structure TrackInformation 154
6.3.4 Énumération SearchState 154
6.3.5 Structure MediaAttributes 154
6.3.6 Structure FindEventResult 154
6.3.7 Structure FindPTZPositionResult 155
6.3.8 Structure PTZPositionFilter 155
6.3.9 Structure MetadataFilter 155
6.3.10 Structure FindMetadataResult 155
6.4 GetRecordingSummary 155
6.5 GetRecordingInformation 156
6.6 GetMediaAttributes 156
6.7 FindRecordings 157
6.8 GetRecordingSearchResults 158
6.9 FindEvents 159
6.10 GetEventSearchResults 160
6.11 FindPTZPosition 161
6.12 GetPTZPositionSearchResults 162
6.13 FindMetadata 163
6.14 GetMetadataSearchResults 164
6.15 EndSearch 165
6.16 GetServiceCapabilities 166
6.17 Descriptions d'événements d'enregistrement 166
6.18 Dialecte XPath 168
7 Contrôle de lecture 168
7.1 Demande d'URI de lecture 168

7.2 ReplayConfiguration 169
7.3 SetReplayConfiguration 169
7.4 GetReplayConfiguration 169
7.5 GetServiceCapabilities 170
8 Restitution 171
8.1 Utilisation du protocole RTSP 171
8.2 Description RTSP 171
8.3 Extension d'en-tête RTP 172
8.3.1 Généralités 172
8.3.2 Horodatages NTP 172
8.3.3 Compatibilité avec l'extension d'en-tête JPEG 173
8.4 Balise de caractéristique RTSP 173
8.5 Lancement de la restitution 173
8.5.1 Généralités 173
8.5.2 Champ d'en-tête Range 174
8.5.3 Champ d'en-tête Rate-Control 175
8.5.4 Champ d'en-tête Frames 175
8.5.5 Points de synchronisation 176
8.6 Lecture inversée 176
8.6.1 Lancement 176
8.6.2 Ordre de transmission de paquets 176
8.6.3 Numéros de séquence RTP 177
8.6.4 Horodatages RTP 178
8.7 RTSP Keepalive 178
8.8 Enregistrement du métrage en cours 178
8.9 Fin de métrage 178
8.10 Go To Time 178
8.11 Utilisation du protocole RTCP 179
9 Format de fichiers d'exportation 179
9.1 Informations collatérales nécessaires 179
9.2 Synchronisation 181
9.3 Correction de l'heure de début 181
9.4 Signature 181
9.4.1 Préparation de l'élément d'entrée de signature 181
9.4.2 Production de la signature 182
9.4.3 Inclusion de la signature produite dans le fichier 182
9.5 Répétition de signature 184
10 Service de récepteur 185
10.1 Généralités 185
10.2 Points de synchronisation 185
10.3 Persistance 185
10.4 Modes du récepteur 185
10.5 Commandes du récepteur 186
10.5.1 GetReceivers 186
10.5.2 GetReceiver 186
10.5.3 CreateReceiver 186
10.5.4 DeleteReceiver 187
10.5.5 ConfigureReceiver 187
10.5.6 SetReceiverMode 188

10.5.7 GetReceiverState 188
10.6 GetServiceCapabilitites 189
10.7 Événements 189
10.7.1 Généralités 189
10.7.2 ChangeState 190
10.7.3 Échec de la connexion 190
Annexe A (informative) Répétition de signature 191
Annexe B (normative) Fichiers de schémas 193
B.1 Contrôle d'enregistrement 193
B.2 Recherche 203
B.3 Contrôle de lecture 210
B.4 Récepteur 212
B.5 Schéma commun 216
Bibliographie 224

Figure 1 – Modèle de stockage avec pistes 123
Figure 2 – Scellage et examen succincts (Source: Wikipédia) 126
Figure 3 – Exemple d'enregistrements et de pistes 128
Figure 4 – Structure RecordingJobConfiguration 130
Figure 5 – Structure RecordingJobStateInformation 141
Figure 6 – Organigramme des états d'enregistrement 153
Figure 7 – Transmission de paquets lors d'une restitution vers l'avant 176
Figure 8 – Transmission de paquets lors d'une restitution inversée 177
Figure A.1 – Présentation de cases en signature simple 191
Figure A.2 – Présentation de cases en signature répétée 191

Tableau 1 – Espaces de noms référencés (avec préfixe) 122
Tableau 2 – Configuration de piste 132
Tableau 3 – Présentation d'un paquet RTP 172
Tableau 4 – Présentation de paquet RTP avec en-tête JPEG 173

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

SYSTÈMES DE VIDÉOSURVEILLANCE DESTINÉS À ÊTRE UTILISÉS DANS LES APPLICATIONS DE SÉCURITÉ –

Partie 2-32: Contrôle d'enregistrement et lecture en fonction des services Web

AVANT-PROPOS

1) La Commission Électrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. À cet effet, l'IEC – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de l'IEC peuvent faire l'objet de droits de brevet. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

La Norme internationale IEC 62676-2-32 a été établie par le comité d'études 79 de l'IEC: Systèmes d'alarme et de sécurité électroniques.

Cette première édition, conjointement avec l'IEC 60839-11-31 et l'IEC 62676-2-31, annule et remplace l'IEC 62676-2-3:2013.

Cette édition inclut les modifications techniques majeures suivantes par rapport à l'IEC 62676-2-3:2013:

a) ajout d'un format de fichiers d'exportation.

Le texte de cette Norme internationale est issu des documents suivants:

FDIS	Rapport de vote
79/621/FDIS	79/623/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette Norme internationale.

Ce document a été rédigé selon les Directives ISO/IEC, Partie 2.

Une liste de toutes les parties de la série IEC 62676, publiées sous le titre général *Systèmes de vidéosurveillance destinés à être utilisés dans les applications de sécurité*, peut être consultée sur le site web de l'IEC.

Le comité a décidé que le contenu de ce document ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous "http://webstore.iec.ch" dans les données relatives au document recherché. À cette date, le document sera

- reconduit,
- supprimé,
- remplacé par une édition révisée, ou
- amendé.

IMPORTANT – Le logo *"colour inside"* qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

INTRODUCTION

Le présent document a pour objet d'assurer une mise en œuvre d'enregistrement et de lecture vidéo en réseau avec interopérabilité complète, constituée de produits provenant de différents fournisseurs. Le présent document décrit le modèle d'enregistrement vidéo en réseau, les interfaces, les types de données et les schémas d'échange de données. Le document réutilise les normes existantes pertinentes lorsqu'elles sont disponibles et présente de nouvelles spécifications, uniquement lorsque cela est nécessaire pour prendre en charge les exigences spécifiques de l'enregistrement et de la lecture vidéo en réseau.

SYSTÈMES DE VIDÉOSURVEILLANCE DESTINÉS À ÊTRE UTILISÉS DANS LES APPLICATIONS DE SÉCURITÉ –

Partie 2-32: Contrôle d'enregistrement et lecture en fonction des services Web

1 Domaine d'application

La présente partie de l'IEC 62676 spécifie l'interface de services web pour la configuration de l'enregistrement des données vidéo et audio, ainsi que des métadonnées. De plus, les événements associés sont définis.

L'Article 4 fournit une définition du modèle de stockage sur lequel se fonde le présent document.

L'utilisation des services web ne relève pas du domaine d'application du présent document. Se reporter à l'IEC 60839-11-31 pour de plus amples informations.

2 Références normatives

Les documents suivants sont cités dans le texte de sorte qu'ils constituent, pour tout ou partie de leur contenu, des exigences du présent document. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

IEC 60839-11-31:2016, *Systèmes d'alarme et de sécurité électroniques – Partie 11-31: Systèmes de contrôle d'accès électronique – Protocole de base d'interopérabilité en fonction des services Web*

IEC 62676-2-31:2019, *Systèmes de vidéosurveillance destinés à être utilisés dans les applications de sécurité – Partie 2-31: Transmission en continu en direct et contrôle en fonction des services web*

Internet Assigned Numbers Authority (IANA), Media Types, *Media Types* [online]. Edited N. Freed et al. [consulté le 2019-02-28]. Disponible sur https://www.iana.org/assignments/media-types/media-types.xhtml

INTERNET ENGINEERING TASK FORCE (IETF). RFC 2326: *Real Time Streaming Protocol (RTSP)* [online]. Edited by H. Schulzrinne et al. April 1998 [consulté le 2019-02-28]. Disponible sur http://www.ietf.org/rfc/rfc2326.txt

INTERNET ENGINEERING TASK FORCE (IETF). RFC 3280, *Internet X.509 Public Key Infrastructure – Certificate and Certificate Revocation List (CRL) Profile* [online]. Edited by Housley, et. al. April 2002 [consulté le 2019-02-28]. Disponible sur http://www.ietf.org/rfc/rfc3280.txt

INTERNET ENGINEERING TASK FORCE (IETF). RFC 3550, *RTP: A Transport Protocol for Real-Time* [online]. Edited by Schulzrinne, et al. Jul 2003 [consulté le 2019-02-28]. Disponible sur https://www.ietf.org/rfc/rfc3550.txt

INTERNET ENGINEERING TASK FORCE (IETF). RFC 4055, *Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure – Certificate and Certificate Revocation List (CRL)* Profile [online]. Edited by Schaad, et al. June 2005 [consulté le 2019-02-28]. Disponible sur https://www.ietf.org/rfc/rfc4055.txt

The World Wide Web Consortium (W3C). SOAP12-PART1, SOAP 1.2 – Part 1, Messaging Framework [online]. Edited by M, Gudgin et al. Apr 2007 {Consulté le 2019-02-28]. Disponible sur https://www.w3.org/TR/soap12-part1/

The World Wide Web Consortium (W3C). XML-Schema 1, *W3C XML Schema – Part 1: Structures Second Edition* [online]. Edited by H. Thompson et al. Oct 2004 [consulté le 2019-02-28]. Disponible sur https://www.w3.org/TR/xmlschema-1/

The World Wide Web Consortium (W3C). XML-Schema 2, *W3C XML Schema – Part 2: Datatypes Second Edition* [online]. Edited by P. Biron et al. Oct 2004 [consulté le 2019-02-28]. Disponible sur https://www.w3.org/TR/xmlschema-2/

The World Wide Web Consortium (W3C). XPath 1.0, *XML Path Language (XPath) Version 1.0* [online]. Edited by J, Clark et al. Nov 1999 [Consulté le 2019-02-28]. Disponible sur https://www.w3.org/TR/1999/REC-xpath-19991116/

Federal Information Processing Standard (FIPS), FIPS 180-4, *Secure Hash Standard (SHS)* [online]. [consulté le 2019-02-28]
Disponible sur https://csrc.nist.gov/publications/detail/fips/180/4/final

3 Termes, définitions et termes abrégés

3.1 Termes et définitions

Pour les besoins du présent document, les termes et définitions suivants s'appliquent.

L'ISO et l'IEC tiennent à jour des bases de données terminologiques destinées à être utilisées en normalisation, consultables aux adresses suivantes:

- IEC Electropedia: disponible à l'adresse http://www.electropedia.org/
- ISO Online browsing platform: disponible à l'adresse http://www.iso.org/obp

3.1.1
unité d'accès
une ou plusieurs trames ou un ou plusieurs échantillons de données audio ou vidéo, ou de métadonnées, contenus dans un groupe de paquets RTP ayant le même temps de présentation

3.1.2
certificat
données associant une clé publique à une entité objet

Note 1 à l'article: Le certificat porte la signature numérique de son émetteur afin de pouvoir vérifier son authenticité.

3.1.3
métadonnées
ensemble des données de transmission en continu, sauf la vidéo et l'audio, comprenant les résultats d'analyse vidéo, les données de position PTZ et d'autres métadonnées (telles que des données textuelles provenant d'applications POS)

3.1.4
enregistrement
conteneur d'un ensemble de pistes audio, vidéo et de métadonnées

Note 1 à l'article: Un enregistrement peut comporter une ou plusieurs pistes. Une piste est perçue comme une chronologie infinie comportant des données à certains moments.

3.1.5
événement d'enregistrement
événement associé à un enregistrement, représenté par un message de notification dans les API

3.1.6
travail d'enregistrement
travail permettant de procéder au transfert des données entre une source de données et un enregistrement particulier à l'aide d'une configuration particulière

3.1.7
signature
signature numérique
schéma de signature numérique
programme mathématique qui permet de démontrer l'authenticité d'un message ou d'un document numérique

3.1.8
piste
voie de données individuelle composée de données vidéo, de données audio et de métadonnées

Note 1 à l'article: Cette définition est conforme à la définition du terme "piste" dans la norme IETF RFC 2326.

3.1.9
analyse vidéo
algorithmes ou programmes utilisés pour analyser les données vidéo et pour générer des données décrivant l'emplacement et le comportement d'objets

3.2 Termes abrégés

CCTV	Closed-Circuit TeleVision (télévision en circuit fermé)
JPEG	Joint Photographic Expert Group (groupe JPEG)
PTZ	Pan Tilt Zoom (panoramique/inclinaison/zoom)
RTCP	RTP Control Protocol (protocole de contrôle en temps réel, protocole RTCP)
RTP	Real -time Transport Protocol (protocole de transmission en temps réel, protocole RTP)
RTSP	Real Time Streaming Protocol (protocole de flux en temps réel, protocole RTSP)
SDP	Session Description Protocol (protocole de description de session, protocole SDP)
SHA	Secure Hash Algorithm (algorithme de hachage sécurisé)
TCP	Transmission Control Protocol (protocole TCP)
UDP	User Datagram Protocol (protocole UDP)
UTC	coordinated universal time (Temps universel coordonné)
UTF	Unicode Transformation Format (format de transformation Unicode)
WSDL	web service description language (langage de description de services web)

4 Vue d'ensemble

4.1 Interfaces

Le présent document fournit un ensemble d'interfaces permettant de prendre en charge les dispositifs de stockage en réseau interopérables, tels que les enregistreurs vidéo en réseau, les enregistreurs vidéo numériques et les caméras à stockage intégré.

Les fonctions suivantes sont prises en charge:

- contrôle d'enregistrement;
- recherche;
- contrôle de lecture.

Ces fonctions sont proposées par trois services liés:

Le service de contrôle d'enregistrement permet à un client de gérer les enregistrements et de configurer le transfert de données entre les sources de données et les enregistrements. La gestion des enregistrements comprend la création et la suppression des enregistrements et des pistes. Le langage WSDL applicable à ce service est spécifié à l'Article B.1.

Le service de recherche, qui permet à un client de rechercher des informations relatives à des enregistrements sur le dispositif de stockage (pour élaborer une vue "chronologique", par exemple) et rechercher des données intéressantes dans un ensemble d'enregistrements. Il s'agit de rechercher des événements inclus dans l'enregistrement de piste de métadonnées. Le langage WSDL applicable à ce service est spécifié à l'Article B.2.

Le service de contrôle de lecture permet à un client de lire les données enregistrées, y compris les données vidéo, les données audio et les métadonnées. Des fonctions permettent de démarrer et d'interrompre la restitution et de modifier la vitesse et le sens du flux restitué. Le service permet également à un client de télécharger des données à partir du dispositif de stockage, de manière à pouvoir offrir une fonction d'exportation. Le langage WSDL applicable à ce service est spécifié à l'Article B.3.

Le présent document comprend également une spécification pour:

- la restitution des données enregistrées;
- un format de fichiers d'exportation;
- un récepteur qui agit comme un point d'extrémité de client RTSP. Le langage WSDL applicable à ce service est spécifié à l'Article B.4.

Le Tableau 1 répertorie les préfixes et espaces de noms utilisés dans la présente spécification. Une annexe est fournie qui correspond aux interfaces définies par le présent document. Les préfixes répertoriés ne font pas partie du présent document, et une mise en œuvre peut utiliser le préfixe de son choix.

Tableau 1 – Espaces de noms référencés (avec préfixe)

Préfixe	URI d'espace de nom	Référence
env	http://www.w3.org/2003/05/soap-envelope	*W3C SOAP12-PART1*
ter	http://www.onvif.org/ver10/error	IEC 60839-11-31
trc	http://www.onvif.org/ver10/recording/wsdl	Article B.1
trp	http://www.onvif.org/ver10/replay/wsdl	Article B.3
trv	http://www.onvif.org/ver10/receiver/wsdl	Article B.4
tse	http://www.onvif.org/ver10/search/wsdl	Article B.2
tt	http://www.onvif.org/ver10/schema	Article B.5
xs	http://www.w3.org/2001/XMLSchema	W3C XML-Schema-1 W3C XML-Schema-2

4.2 Modèle de stockage

Les interfaces de stockage de la présente norme présentent une vue logique des données sur le dispositif de stockage. Cette vue est totalement indépendante de la manière dont les données peuvent être physiquement stockées sur le disque.

Le concept clé du modèle de stockage est l'enregistrement. Le terme "enregistrement" est utilisé dans la présente spécification pour indiquer un conteneur d'un ensemble de pistes audio, vidéo et de métadonnées associées, provenant en général de la même source de données (une caméra, par exemple). Un enregistrement peut comporter un certain nombre de pistes. Une piste est perçue comme une chronologie infinie comportant des données à certains moments.

Un enregistrement est au moins capable de comporter trois pistes: une piste audio, une piste vidéo et une piste de métadonnées. Certaines mises en œuvre du service d'enregistrement peuvent prendre en charge plusieurs pistes de chaque type. Par exemple, le même enregistrement peut comporter deux pistes vidéo, l'une contenant un flux basse résolution ou bas débit, et l'autre un flux haute résolution ou haut débit; voir la Figure 1.

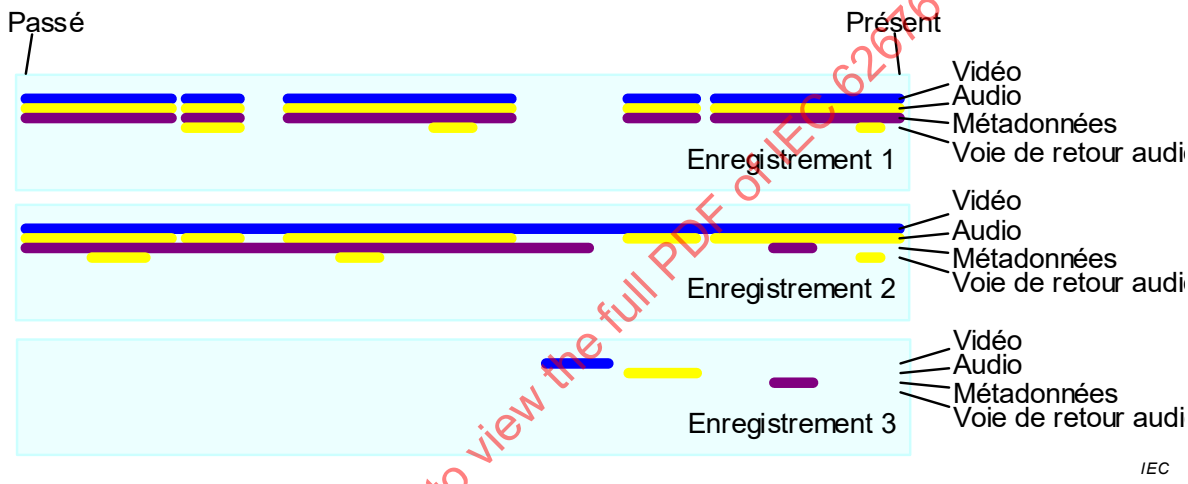

Figure 1 – Modèle de stockage avec pistes

Il est important de noter que les interfaces de stockage n'exposent pas les structures de stockage internes sur le dispositif. En particulier, un enregistrement n'est pas destiné à représenter un seul fichier du disque même si, dans de nombreuses mises en œuvre de dispositif de stockage, un enregistrement est physiquement stocké dans une série de fichiers. Par exemple, certaines mises en œuvre de caméra enregistrent des alarmes en créant un fichier distinct pour chaque alarme déclenchée. Bien que chaque fichier puisse être représenté comme étant un enregistrement différent, le modèle décrit dans la présente norme a pour objet de regrouper tous ces fichiers dans un seul enregistrement.

Dans un enregistrement, les zones dans lesquelles les données sont réellement enregistrées sont représentées par des paires d'événements, chacune d'elles étant composée d'un événement de début d'enregistrement et d'un événement de fin d'enregistrement. Un client peut élaborer la vue logique des enregistrements à l'aide des méthodes FindRecordings et FindEvents du service de recherche.

Si les métadonnées sont enregistrées, la piste de métadonnées peut contenir tous les événements générés par la source de données (voir Article 10 de l'IEC 60839-11-31:2016 et 6.3.8 de l'IEC 62676-2-31:2019. De plus, d'un point de vue conceptuel, un dispositif enregistre également les événements d'historique définis par le présent document (voir 6.17), cela incluant des informations telles que le début et la fin d'une plage de données enregistrée. D'un point de vue conceptuel, un dispositif peut également enregistrer les événements d'historique spécifiques au fournisseur. Les événements générés par le dispositif ne sont pas insérés dans les pistes de métadonnées existantes des enregistrements. La méthode FindEvents du service de recherche peut trouver tous les événements enregistrés.

4.3 Contrôle d'enregistrement

Le service d'enregistrement permet à un client de gérer les enregistrements et de configurer le transfert de données entre les sources de données et les enregistrements. La gestion des enregistrements comprend la création et la suppression des enregistrements et des pistes.

Les travaux d'enregistrement permettent de transférer des données d'une source d'enregistrement vers un enregistrement. Une source d'enregistrement peut être un objet de récepteur créé avec le service de récepteur, ou un profil multimédia qui code les données sur un dispositif local. Le profil multimédia peut être utilisé comme une source sur une caméra à stockage intégré.

Pour sauvegarder des données sur un enregistrement, un client crée en premier lieu un enregistrement et vérifie que l'enregistrement dispose des pistes nécessaires. Ensuite, le client crée un travail d'enregistrement qui extrait les données d'une ou de plusieurs sources et les stocke sur les pistes de l'enregistrement.

Les clients peuvent configurer plusieurs travaux d'enregistrement dans le même enregistrement. Si plusieurs travaux d'enregistrement sont actifs, le dispositif utilise un schéma de priorité pour faire un choix parmi les pistes définies dans les travaux d'enregistrement. Les clients peuvent à tout moment modifier le mode des travaux d'enregistrement, permettant ainsi la mise en œuvre de caractéristiques telles que l'enregistrement d'alarme ou l'enregistrement manuel.

Le travail d'enregistrement s'appuie sur le service de récepteur pour recevoir des données provenant d'autres dispositifs au moyen des objets de récepteur identifiés par ReceiverTokens.

4.4 Recherche

Le service de recherche permet à un client de rechercher des informations relatives à des enregistrements sur le dispositif de stockage (pour élaborer une vue "chronologique", par exemple) et rechercher des données intéressantes dans un ensemble d'enregistrements. Il s'agit de rechercher des événements et d'autres informations inclus dans l'enregistrement de piste de métadonnées.

Le service de recherche offre les fonctionnalités suivantes:

- recherche d'enregistrements et d'informations relatives à chaque enregistrement;
- recherche d'événements dans les métadonnées et parmi les événements historiques;
- recherche de positions PTZ dans les métadonnées;
- recherche d'autres informations dans les métadonnées (texte issu de systèmes de point de vente électronique (EPOS – *electronic point-of-sale*, par exemple).

La recherche réelle, qui est asynchrone, a lieu sous la forme d'opérations couplées de recherche et de résultats. Chaque opération de recherche lance une session de recherche. Le client peut alors obtenir les résultats issus de la session de recherche par incréments ou en totalité, selon la mise en œuvre et l'étendue de la recherche. Il existe quatre paires d'opérations de recherche pour les enregistrements, les événements d'enregistrement, les positions PTZ et les métadonnées:

- FindRecordings et GetRecordingSearchResults;
- FindEvents et GetEventSearchResults;
- FindPTZPosition et GetPTZPositionSearchResults;
- FindMetadata et GetMetadataSearchResults.

4.5 Contrôle de lecture

Le service de contrôle de lecture offre un mécanisme de lecture des données vidéo, des données audio et des métadonnées. Ce mécanisme peut également être utilisé pour télécharger des données à partir du dispositif de stockage, de manière à pouvoir offrir une fonction d'exportation.

Le protocole de lecture repose sur le protocole RTSP (IETF RFC 2326). Toutefois, le protocole RTSP ne prenant pas directement en charge toutes les exigences en matière de lecture, plusieurs extensions lui ont été ajoutées. En particulier, une extension d'en-tête RTP est définie pour pouvoir associer un horodatage absolu à chaque unité d'accès (une trame vidéo, par exemple) et acheminer des informations relatives à la continuité du flux.

La commande GetReplayUri du service de lecture retourne l'URL RTSP d'un enregistrement afin de pouvoir assurer la lecture à l'aide du protocole RTSP.

4.6 Format de fichiers d'exportation

4.6.1 Présentation

Toutes les données qu'un utilisateur souhaite acheminer séparément sont placées dans un contenant métaphorique. Le contenant est ensuite scellé afin de détecter toute violation. Quiconque souhaite utiliser les données du contenant examine en premier lieu le joint étanche. Les données du contenant sont identiques aux données d'origine tant que le joint étanche est intact, voir la Figure 2.

Ici, le contenant métaphorique est représenté par un fichier et le joint étanche est représenté par une signature pour toutes les données du fichier.

L'approche "à preuves multiples" repose sur des procédures applicables aux données multimédias et aux métadonnées associées à extraire en toute sécurité d'un emplacement de stockage fiable dans un fichier distinct. Cette approche définit les métadonnées qui doivent être protégées afin d'assurer une lecture exacte. Les données sont fournies "telles qu'elles" sans assertion complémentaire, de quelque nature que ce soit, afin de maintenir la démonstration.

Les décharges de puissance de traitement comprennent l'utilisation de fonctions de hachage avant l'application des algorithmes de signature. Plusieurs niveaux de signatures peuvent être appliqués afin de recueillir les informations complémentaires dans un seul fichier étanche.

Des normes internationales relatives à l'état de la technique sont appliquées pour la structure du fichier, ainsi que les algorithmes de hachage et de signature. Le format d'application de surveillance défini par l'ISO/IEC 23000-10 et la signature RSA2048, ainsi que l'algorithme de hachage SHA-256 agréé par le NIST sont mis en œuvre pour l'interopérabilité la plus répandue.

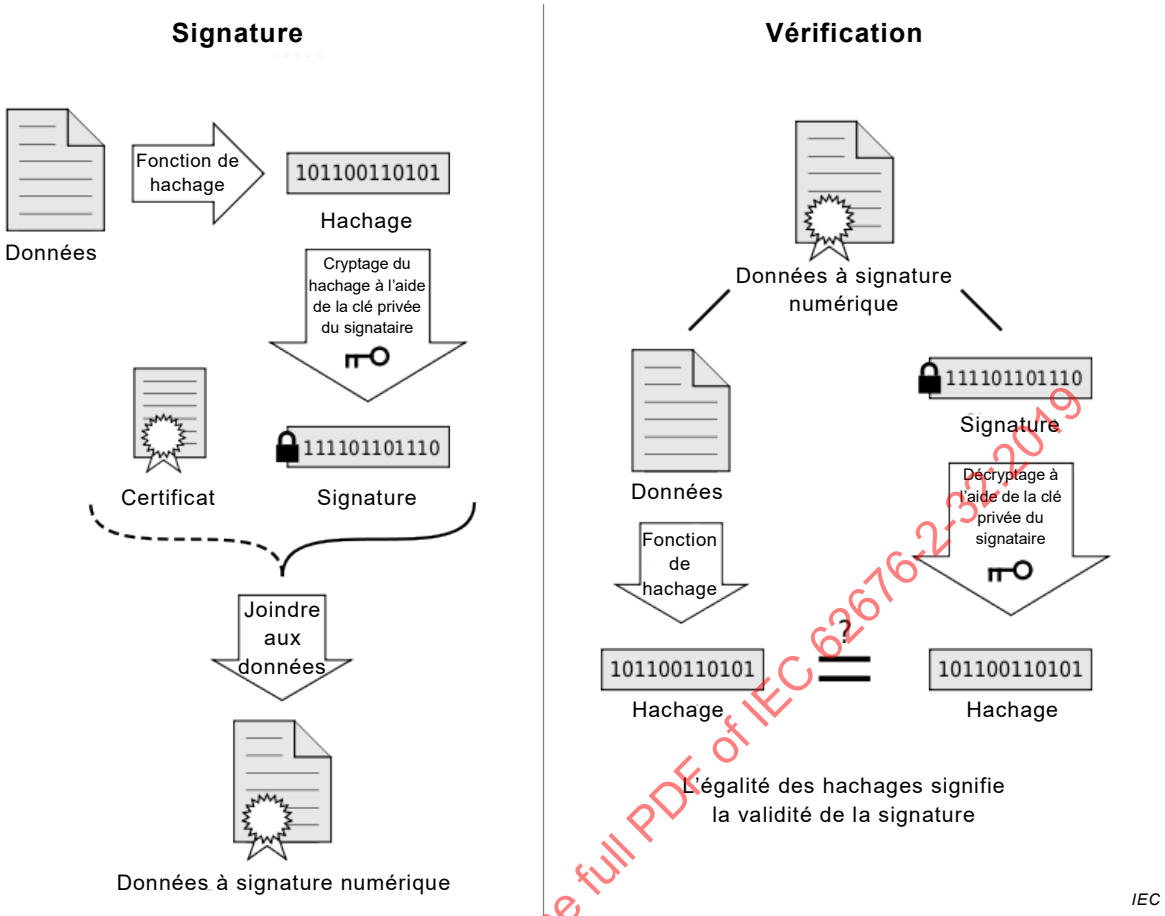

Figure 2 – Scellage et examen succincts (Source: Wikipédia)

4.6.2 Cas d'utilisation 1: Restitution de vidéoclips morcelés et surdimensionnés à distance

Un opérateur exporte un vidéoclip avec des données audio associées provenant d'un enregistreur vidéo numérique de marque A sur deux DVD, parce qu'il ne pouvait pas être contenu sur un seul DVD. La période d'enregistrement sélectionnée comporte des écarts étant donné que l'enregistreur ne fonctionnait qu'en cas de détection d'un mouvement. Les DVD sont ensuite transmis vers un second site comportant un logiciel où leur contenu est copié sur un disque dur local. L'utilisateur le restitue alors dans le système de gestion des vidéos de marque B. L'opérateur au poste de restitution souhaite observer les écarts d'enregistrement, et rechercher chaque moment d'exportation d'une vidéo. Au moment de la restitution, il s'attend à ce que la lecture de la vidéo s'effectue sans difficulté avec une restitution audio à synchronisation labiale.

4.6.3 Cas d'utilisation 2: Analyse criminalistique dans un tribunal

Un tribunal reçoit des vidéoclips provenant d'une épicerie, d'un système de vidéosurveillance publique et d'un opérateur de métropolitain. Les trois vidéos sont toutes visionnées sur le lecteur vidéo agréé par le tribunal.

Les juges souhaitent voir le suspect dans les trois situations avec les informations temporelles exactes. Ils souhaitent également disposer d'informations relatives au moment d'exportation des vidéoclips, ainsi que le caractère exhaustif et authentique ou non de la séquence vidéo.

4.6.4 Cas d'utilisation 3: Restitution avec des lecteurs non équipés selon la présente spécification

Une personne autorisée reçoit des vidéoclips au format défini dans la présente spécification et souhaite lire les données multimédias sur des lecteurs conformes aux définitions normalisées fondamentales. L'interprétation des informations complémentaires ajoutées par la présente spécification n'est pas exigée.

4.7 Récepteur

Un récepteur est un objet qui agit comme un point d'extrémité de client RTSP. Les récepteurs sont utilisés par d'autres services qui utilisent des flux multimédias (les services d'enregistrement et autres services). La configuration d'un récepteur détermine le point d'extrémité RTSP vers lequel il convient de se connecter, ainsi que les paramètres de connexion qu'il convient d'utiliser.

Un récepteur peut fonctionner en trois modes distincts:

- AlwaysConnect: Le récepteur tente de maintenir une connexion permanente au point d'extrémité configuré.
- NeverConnect: Le récepteur ne tente pas de se connecter.
- AutoConnect: Le récepteur se connecte à la demande des utilisateurs des flux multimédia.

Un seul récepteur peut être utilisé par plusieurs utilisateurs. Par exemple, pour enregistrer un flux et l'analyser, un travail d'enregistrement et un moteur d'analyses peuvent être associés au même récepteur. Si le récepteur utilise le mode "Connexion automatique" (*Auto Connect*), il se connecte à chaque fois que le travail d'enregistrement ou le moteur d'analyses est actif, et se déconnecte lorsqu'ils sont inactifs.

Les récepteurs peuvent être créés et supprimés soit manuellement, en appelant les opérations CreateReceiver et DeleteReceiver du service de récepteur, soit automatiquement par d'autres services. Par exemple, si un travail d'enregistrement est créé avec l'option "AutoCreateReceiver", il est automatiquement créé et associé à un récepteur. La suppression d'un travail d'enregistrement supprime également le récepteur.

5 Service de contrôle d'enregistrement

5.1 Vue d'ensemble

Le service d'enregistrement permet à un client de gérer les enregistrements et de configurer le transfert de données entre les sources de données et les enregistrements. La gestion des enregistrements comprend la création et la suppression des enregistrements et des pistes, ainsi que le verrouillage et le déverrouillage des plages d'enregistrements et la suppression des données enregistrées.

Les travaux d'enregistrement permettent de transférer des données d'une source d'enregistrement vers un enregistrement. Une source d'enregistrement peut être un objet de récepteur créé avec le service de récepteur, ou un profil multimédia qui code les données sur un dispositif local. Le profil multimédia peut être utilisé comme une source sur une caméra à stockage intégré.

Le terme "enregistrement" est utilisé dans la présente spécification pour indiquer un conteneur d'un ensemble de pistes audio, vidéo et de métadonnées. Un enregistrement peut comporter un certain nombre de pistes. Une piste est perçue comme une chronologie infinie comportant des données à certains moments.

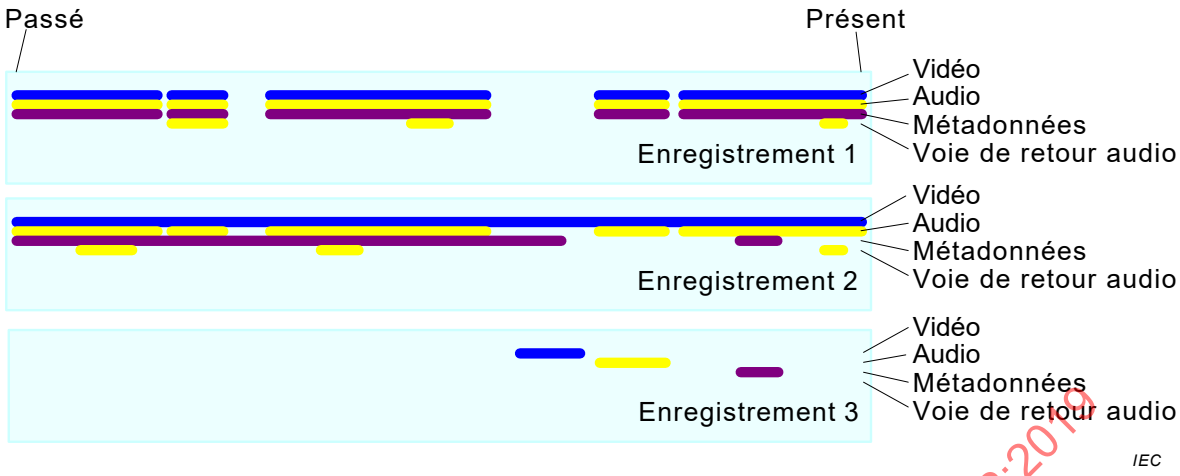

Figure 3 – Exemple d'enregistrements et de pistes

La Figure 3 présente trois enregistrements, chacun avec une piste vidéo, une piste de métadonnées et deux pistes audio. Une deuxième piste audio est utilisée dans ce cas pour le stockage de la voie de retour audio.

Un enregistrement doit être au moins capable de comporter trois pistes: une piste audio, une piste vidéo et une piste de métadonnées. Certaines mises en œuvre du service d'enregistrement peuvent prendre en charge plusieurs pistes de chaque type. Toutes les données enregistrées d'une piste doivent avoir le même codage.

Pour sauvegarder des données sur un enregistrement, un client crée en premier lieu un enregistrement et vérifie que l'enregistrement dispose des pistes nécessaires. Ensuite, le client crée un travail d'enregistrement qui extrait les données d'une ou de plusieurs sources et les stocke sur les pistes de l'enregistrement.

Les clients peuvent configurer plusieurs travaux d'enregistrement dans le même enregistrement. Si plusieurs travaux d'enregistrement sont actifs, le dispositif utilise un schéma de priorité pour faire un choix parmi les pistes définies dans les travaux d'enregistrement. Les clients peuvent à tout moment modifier le mode des travaux d'enregistrement, en permettant la mise en œuvre de caractéristiques telles que l'enregistrement d'alarme ou l'enregistrement manuel.

Le travail d'enregistrement s'appuie sur le service de récepteur pour recevoir des données provenant d'autres dispositifs au moyen des objets de récepteur identifiés par ReceiverTokens.

Dans les cas où un client utilise un objet de récepteur avec un seul travail d'enregistrement, le service d'enregistrement peut créer et supprimer automatiquement les objets de récepteur. La création automatique est indiquée par le drapeau AutoCreateReceiver dans la structure de configuration du travail d'enregistrement. Les objets de récepteur créés de cette manière doivent être automatiquement supprimés si plus aucun travail d'enregistrement ne les utilise. Des valeurs vides doivent être attribuées à tous les champs d'un objet de récepteur automatiquement créé. Il convient que le client configure l'objet de récepteur après avoir créé le travail d'enregistrement.

Cette vue normalisée des enregistrements est une vue logique indépendante de la manière dont les enregistrements sont physiquement stockés sur le disque. Par exemple, certaines mises en œuvre de caméra enregistrent des alarmes en créant un fichier distinct sur un système de fichiers FAT pour chaque alarme déclenchée. Bien que chaque fichier puisse être représenté comme étant un enregistrement différent tel que défini par le présent document, le modèle décrit dans le présent document a pour objet de regrouper tous ces fichiers dans un seul enregistrement. En recherchant l'événement "DataPresent" avec la méthode FindEvents du

service de recherche, un client peut déterminer les heures de début de l'enregistrement vidéo et à quel moment il s'est arrêté.

Si les métadonnées sont enregistrées, la piste de métadonnées peut contenir tous les événements générés par la source de données (voir Article 10 de l'IEC 60839-11-31:2016 et 6.3.8 de l'IEC 62676-2-31:2019. De plus, d'un point de vue conceptuel, un dispositif enregistre également les événements d'historique définis par le présent document (voir 6.17), cela incluant des informations telles que le début et la fin d'une plage de données enregistrée. D'un point de vue conceptuel, un dispositif peut également enregistrer les événements d'historique spécifiques au fournisseur. Les événements générés par le dispositif ne sont pas insérés dans les pistes de métadonnées existantes des enregistrements. La méthode FindEvents du service de recherche peut trouver tous les événements enregistrés.

De nombreuses mises en œuvre de dispositif suppriment automatiquement les données enregistrées les plus anciennes du stockage afin de libérer de l'espace pour les nouveaux enregistrements. Les verrous offrent un mécanisme permettant à l'utilisateur de sélectionner des plages de données. Une plage de données verrouillée n'est pas automatiquement supprimée. La prise en charge des verrous est réservée aux versions ultérieures de la spécification.

5.2 Exigences générales

Tous les objets créés dans le service d'enregistrement doivent être persistants, c'est-à-dire qu'ils doivent résister à un cycle d'alimentation. De même, toutes les données de configuration des objets doivent être persistantes.

5.3 Structures de données

5.3.1 RecordingConfiguration

La structure RecordingConfiguration doit être utilisée pour configurer les enregistrements avec CreateRecordings et Get/SetRecordingConfiguration.

MaximumRetentionTime spécifie la durée maximale pendant laquelle les données d'une piste quelconque de l'enregistrement doivent être stockées. Le dispositif doit supprimer toutes les données plus anciennes que le temps de rétention maximal. Ces données ne doivent plus être accessibles. Si la valeur 0 est attribuée à MaximumRetentionPeriod, le dispositif ne doit pas limiter le temps de rétention des données stockées, sauf par des contraintes de ressource. Quelle que soit la valeur de MaximumRetentionTime, le dispositif peut automatiquement supprimer des enregistrements pour libérer de l'espace de stockage pour de nouveaux enregistrements.

Aucun des autres champs définis dans cette structure ne doit être utilisé par le dispositif. La structure stocke simplement ces informations et doit les retourner au moyen des méthodes GetRecordingConfiguration et GetRecordingInformation.

Un dispositif peut tronquer toute chaîne descriptive sans générer de défaut en cas de dépassement de la longueur prise en charge. Les chaînes descriptives sont les suivantes: Location (Emplacement), Description et Content (Contenu).

5.3.2 TrackConfiguration

La structure TrackConfiguration doit être utilisée pour configurer les pistes avec CreateTrack et Get/SetTrackConfiguration.

TrackConfiguration contient les champs suivants:

TrackType définit le type de données de la piste. Il doit être égal aux chaînes "Video" ("Vidéo"), "Audio" ou "Metadata" ("Métadonnées"). La piste doit uniquement pouvoir contenir des données de ce type.

Aucun des autres champs définis dans cette structure ne doit être utilisé par le dispositif. La structure stocke simplement ces informations et doit les retourner au moyen des méthodes GetTrackConfiguration et GetRecordingInformation.

5.3.3 RecordingJobConfiguration

La structure RecordingJobConfiguration doit contenir la configuration d'un travail d'enregistrement. RecordingJobConfiguration, en tant que schéma UML, peut être perçue telle que représentée à la Figure 4.

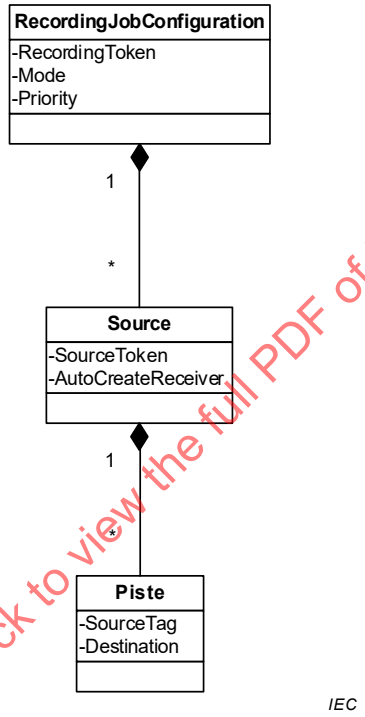

Figure 4 – Structure RecordingJobConfiguration

RecordingJobConfiguration contient les champs suivants:

RecordingToken: Identifie l'enregistrement dans lequel ce travail doit stocker les données reçues.

Mode: S'il s'agit du mode "Idle" ("Veille"), il ne doit rien se passer. S'il s'agit du mode "Active" ("Actif") et que le travail d'enregistrement présente la priorité la plus élevée, le dispositif doit tenter d'obtenir les données auprès des récepteurs. Un client doit utiliser GetRecordingJobState pour déterminer si le transfert de données a réellement lieu. Les seules valeurs valides de Mode doivent être "Idle" et "Active".

Priority: Il doit s'agir d'un nombre non négatif. Si plusieurs travaux d'enregistrement stockent des données sur la même piste, le dispositif stocke uniquement les données dont le travail d'enregistrement présente la priorité la plus élevée. La priorité est spécifiée par travail d'enregistrement, mais le dispositif doit déterminer la priorité de chaque piste individuellement. Si deux travaux d'enregistrement présentent la même priorité, le dispositif doit enregistrer les données correspondant au travail d'enregistrement qui a été activé en dernier.

La valeur 0 indique la priorité la plus basse. Des valeurs plus élevées doivent indiquer une priorité plus élevée.

SourceToken: Ce champ doit être une référence à la source des données. Le type de la source est déterminé par l'attribut Type de la structure SourceToken. La valeur est donnée sous la forme d'une URI[1]. Le présent document définit les deux valeurs ci-dessous pour l'attribut Type. Si le Type est http://www.onvif.org/ver10/schema/Receiver, le jeton est un ReceiverReference (voir 10). Dans ce cas, le dispositif doit recevoir les données sur le réseau. Si le Type est http://www.onvif.org/ver10/schema/Profile, le jeton identifie un profil multimédia (voir l'IEC 62676-11-31), demandant au dispositif d'obtenir les données auprès d'un profil qui existe sur le dispositif local.

Un dispositif qui comprend la prise en charge du service multimédia tel que défini dans l'IEC 62676-11-31 doit prendre en charge un jeton de profil multimédia et un dispositif qui comprend la prise en charge du service de récepteur tel que défini dans la présente norme doit prendre en charge un jeton Receiver.

Si SourceToken est ignoré, AutoCreateReceiver doit être vrai.

AutoCreateReceiver: Si la valeur de ce champ est TRUE, et que SourceToken est ignoré, le dispositif doit créer un objet de récepteur (avec le service de récepteur) et attribuer ReceiverReference au champ SourceToken. Lors de l'extraction de RecordingJobConfiguration du dispositif, le champ AutoCreateReceiver ne doit jamais être présent.

SourceTag: Si le flux RTSP reçu contient plusieurs pistes de même type, SourceTag fait la distinction entre elles.

Destination: La destination est le jeton de la piste sur laquelle le dispositif doit stocker les données reçues. Toutes les pistes doivent appartenir à l'enregistrement identifié par le RecordingToken.

Le champ TrackInformation d'une piste contient une source unique. Dans le cas où plusieurs RecordingJobs avec une source différente effectuent un enregistrement sur la même piste, le travail d'enregistrement consigné dans les TrackInformation correspondantes de l'API de RecordingSearch n'est pas identifié.

5.4 CreateRecording

CreateRecording doit créer un nouvel enregistrement. Ce nouvel enregistrement doit être créé avec une piste pour chaque TrackType pris en charge (voir 5.21).

Cette méthode est facultative. Elle doit être disponible si la valeur de la fonctionnalité Recording/DynamicRecordings est TRUE.

DEMANDE:

RecordingConfiguration [tt:RecordingConfiguration]
Contient la configuration initiale de l'enregistrement.

RÉPONSE:

RecordingToken [tt:RecordingReference]
Référence à l'enregistrement créé.

[1] L'URI n'est utilisée que pour créer un identifiant unique, elle ne sert à extraire aucun document.

DÉFAUTS:

env:Receiver – ter:Action – ter:MaxRecordings
Le dispositif ne peut pas créer un nouvel enregistrement, car il a déjà atteint le nombre maximal d'enregistrements qu'il prend en charge.

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
RecordConfiguration n'est pas valide.

env:Receiver – ter:ActionNotSupported – ter:NotImplemented
Cette méthode facultative n'est pas mise en œuvre.

CLASSE D'ACCÈS:

ACTUATE

S'il a été exécuté avec succès, CreateRecording doit avoir créé trois pistes avec les configurations présentées dans le Tableau 2.

Tableau 2 – Configuration de piste

TrackToken	TrackType
VIDEO001	Vidéo
AUDIO001	Audio
META001	Métadonnées

La valeur 0 (illimité) doit être attribuée à l'élément MaximumRetentionTime de la RecordingConfigurations, et une chaîne vide doit être attribuée à l'élément Description de toutes les TrackConfigurations.

5.5 DeleteRecording

DeleteRecording doit supprimer un objet d'enregistrement. À chaque fois qu'un enregistrement est supprimé, le dispositif doit supprimer toutes les pistes qui font partie de l'enregistrement et doit supprimer tous les travaux d'enregistrement. Pour chaque travail d'enregistrement supprimé, le dispositif doit également supprimer tous les objets de récepteur associés au travail d'enregistrement automatiquement créés à l'aide du champ AutoCreateReceiver de la structure de configuration du travail d'enregistrement, et qui ne sont pas utilisés dans un autre travail d'enregistrement.

Cette méthode est facultative. Elle doit être disponible si la valeur de la fonctionnalité Recording/DynamicRecordings est TRUE.

DEMANDE:

RecordingToken [tt:RecordingReference]
Identifie l'enregistrement qui doit être supprimé.

RÉPONSE:

Ceci est un message vide.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
RecordingToken ne fait pas référence à un enregistrement existant.

env:Receiver – ter:ActionNotSupported – ter:NotImplemented
Le dispositif ne peut pas supprimer les enregistrements.

env:Receiver – ter:Action – ter:CannotDelete
Cet enregistrement spécifique ne peut pas être supprimé.

CLASSE D'ACCÈS:

ACTUATE

5.6 GetRecordings

GetRecordings doit retourner une description de tous les enregistrements dans le dispositif. Cette description doit inclure une liste de toutes les pistes pour chaque enregistrement.

DEMANDE:

Ceci est un message vide.

RÉPONSE:

RecordingItem – facultatif, non limité [tt:GetRecordingsResponseItem]
Identifie un enregistrement et sa configuration actuelle

DÉFAUTS:

Aucun

CLASSE D'ACCÈS:

READ_MEDIA

5.7 SetRecordingConfiguration

SetRecordingConfiguration doit modifier la configuration d'un enregistrement.

DEMANDE:

RecordingToken [RecordingToken]
Identifie l'enregistrement qui doit être modifié.

RecordingConfiguration [RecordingConfiguration]
Nouvelle configuration de l'enregistrement.

RÉPONSE:

Ceci est un message vide.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter: BadConfiguration
La configuration n'est pas valide.

env:Sender – ter:InvalidArgVal – ter:NoRecording
RecordingToken ne fait pas référence à un enregistrement existant.

CLASSE D'ACCÈS:

ACTUATE

5.8 GetRecordingConfiguration

GetRecordingConfiguration doit extraire la configuration d'un enregistrement.

DEMANDE:

RecordingToken [tt:RecordingReference]
Identifie l'enregistrement dont la configuration doit être extraite.

RÉPONSE:

RecordingConfiguration [tt:RecordingConfiguration]
Configuration actuelle de l'enregistrement demandé.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
RecordingToken ne fait pas référence à un enregistrement existant.

CLASSE D'ACCÈS:

READ_MEDIA

5.9 CreateTrack

Cette méthode doit créer une nouvelle piste dans un enregistrement si la méthode GetRecordingOptions indique les pistes de rechange pour l'enregistrement. Il est nécessaire de définir le SpareXXX (où XXX est le type de piste) dans le cas d'une piste à créer.

Cette méthode est facultative. Elle doit être disponible si la valeur de la fonctionnalité Recording/DynamicTracks est TRUE.

DEMANDE:

RecordingToken [tt:RecordingReference]
Identifie l'enregistrement auquel une piste doit être ajoutée.

TrackConfiguration [tt:TrackConfiguration]
Configuration de la nouvelle piste.

RÉPONSE:

TrackToken [tt:TrackReference]
Identifie la piste nouvellement créée. Un dispositif doit garantir que le TrackToken est unique dans l'enregistrement auquel appartient la nouvelle piste.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
RecordingToken ne fait pas référence à un enregistrement existant.

env:Receiver – ter:Action – ter:MaxTracks
La nouvelle piste ne peut pas être créée, car le nombre maximal de pistes que le dispositif prend en charge pour cet enregistrement a été atteint.

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
TrackConfiguration n'est pas valide.

env:Receiver – ter:ActionNotSupported – ter:NotImplemented
Cette méthode facultative n'est pas mise en œuvre.

CLASSE D'ACCÈS:

ACTUATE

Un TrackToken en lui-même n'identifie pas de manière unique une piste particulière. Les pistes de différents enregistrements peuvent avoir le même TrackToken.

5.10 DeleteTrack

DeleteTrack doit supprimer une piste d'un enregistrement. Toutes les données de la piste doivent être supprimées.

Cette méthode est facultative. Elle doit être disponible si la valeur de la fonctionnalité Recording/DynamicTracks est TRUE.

DEMANDE:

RecordingToken [tt:RecordingReference]
Identifie l'enregistrement duquel supprimer la piste.

TrackToken [tt:TrackReference]
Identifie la piste à supprimer.

RÉPONSE:

Ceci est un message vide.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
RecordingToken ne fait pas référence à un enregistrement existant.

env:Sender – ter:InvalidArgVal – ter:NoTrack
TrackToken ne fait pas référence à une piste existante de l'enregistrement.

env:Receiver – ter:Action – ter:CannotDelete
Cette piste spécifique ne peut pas être supprimée

env:Receiver – ter:ActionNotSupported – ter:NotImplemented
Cette méthode facultative n'est pas mise en œuvre.

CLASSE D'ACCÈS:

ACTUATE

5.11 GetTrackConfiguration

GetTrackConfiguration doit extraire la configuration d'une piste spécifique.

DEMANDE:

RecordingToken [tt:RecordingReference]
Identifie l'enregistrement.

TrackToken [tt:TrackReference]
Identifie la piste dans l'enregistrement duquel obtenir la configuration de piste.

RÉPONSE:

TrackConfiguration [tt:TrackConfiguration]
Configuration actuelle de la piste.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
RecordingToken ne fait pas référence à un enregistrement existant.

env:Sender – ter:InvalidArgVal – ter:NoTrack
TrackToken ne fait pas référence à une piste existante de l'enregistrement.

CLASSE D'ACCÈS:

READ_MEDIA

5.12 SetTrackConfiguration

SetTrackConfiguration doit modifier la configuration d'une piste. TrackType doit être ignoré par le dispositif étant donné qu'il ne peut pas être modifié. TrackConfiguration est la nouvelle configuration de la piste.

DEMANDE:

RecordingToken [tt:RecordingReference]
Identifie l'enregistrement.

TrackToken [tt:TrackReference]
Identifie l'enregistrement dans la piste à partir de laquelle définir la configuration de piste.

TrackConfiguration [tt:TrackConfiguration]
Nouvelle configuration de la piste.

RÉPONSE:

Ceci est un message vide.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
RecordingToken ne fait pas référence à un enregistrement existant.

env:Sender – ter:InvalidArgVal – ter:NoTrack
TrackToken ne fait pas référence à une piste existante de l'enregistrement.

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
Le contenu de l'objet de configuration n'est pas valide.

CLASSE D'ACCÈS:

ACTUATE

5.13 CreateRecordingJob

CreateRecordingJob doit créer un nouveau travail d'enregistrement. Un dispositif doit prendre en charge l'ajout d'un RecordingJob à un enregistrement pour lequel il indique les travaux de rechange (Spare) au moyen de la commande GetRecordingOptions.

DEMANDE:

JobConfiguration [tt:RecordingJobConfiguration]
Configuration du nouveau travail d'enregistrement.

RÉPONSE:

JobToken [tt:RecordingJobReference]
Identifie le travail d'enregistrement créé.

JobConfiguration [tt:RecordingJobConfiguration]
Configuration telle qu'utilisée par le dispositif. Elle peut être différente de la structure JobConfiguration transmise à CreateRecordingJob.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
Le RecordingToken indiqué dans la structure JobConfiguration ne fait pas référence à un enregistrement existant.

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
Le contenu de JobConfiguration n'est pas valide.

env:Receiver – ter:Action – ter:MaxRecordingJobs
Le nombre maximal de travaux d'enregistrement que le dispositif peut traiter a été atteint.

env:Receiver – ter:Action – ter:MaxReceivers
Si la valeur TRUE est attribuée au drapeau AutoCreateReceivers, cette erreur peut être retournée si le service de récepteur ne peut pas créer un nouveau récepteur.

CLASSE D'ACCÈS:

ACTUATE

L'élément JobConfiguration retourné depuis CreateRecordingJob doit être identique à l'élément JobConfiguration transmis dans CreateRecordingJob, sauf pour ReceiverToken et AutoCreateReceiver. Dans la structure retournée, ReceiverToken doit être présent et valide, et le champ AutoCreateReceiver doit être ignoré.

5.14 DeleteRecordingJob

DeleteRecordingJob supprime un travail d'enregistrement. Il doit également supprimer de manière implicite tous les objets de récepteur associés au travail d'enregistrement et automatiquement créés à l'aide du champ AutoCreateReceiver de la structure de configuration du travail d'enregistrement, et qui ne sont pas utilisés dans un autre travail d'enregistrement.

DEMANDE:

JobToken [tt:RecordingJobReference]
Identifie le travail d'enregistrement qui doit être supprimé.

RÉPONSE:

Ceci est un message vide.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:NoRecordingJob
JobToken ne fait pas référence à un travail existant.

CLASSE D'ACCÈS:

ACTUATE

5.15 GetRecordingJobs

GetRecordingJobs doit retourner une liste de tous les travaux d'enregistrement dans le dispositif.

DEMANDE:

Ceci est un message vide.

RÉPONSE:

JobItem– facultatif, non limité [tt:GetRecordingJobsResponseItem]
Identifie un travail dans le dispositif et contient sa configuration actuelle.

DÉFAUTS:

Aucun

CLASSE D'ACCÈS:

READ_MEDIA

5.16 SetRecordingJobConfiguration

SetRecordingJobConfiguration doit modifier la configuration d'un travail d'enregistrement. Un dispositif doit refuser une demande qui tente de modifier RecordingToken.

L'élément JobConfiguration retourné par un dispositif depuis SetRecordingJobConfiguration doit être identique à l'élément JobConfiguration transmis dans SetRecordingJobConfiguration, sauf pour ReceiverToken et AutoCreateReceiver. Dans la structure retournée, ReceiverToken doit être présent et valide, et le champ AutoCreateReceiver doit être ignoré.

DEMANDE:

JobToken [tt:RecordingJobReference]
Identifie le travail d'enregistrement à mettre à jour.

JobConfiguration [tt:RecordingJobConfiguration]
Configuration à appliquer au travail d'enregistrement.

RÉPONSE:

JobConfiguration [tt:RecordingJobConfiguration]
La structure JobConfiguration doit être la configuration telle qu'elle est utilisée par le dispositif. Elle peut être différente de la structure JobConfiguration transmise à SetRecordingJobConfiguration.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:NoRecordingJob
JobToken ne fait pas référence à un travail existant.

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
Le contenu de JobConfiguration n'est pas valide.

env:Receiver – ter:Action – ter:MaxReceivers
Si la valeur TRUE est attribuée au drapeau AutoCreateReceivers, cette erreur peut être retournée si le service de récepteur ne peut pas créer un nouveau récepteur.

CLASSE D'ACCÈS:

ACTUATE

SetRecordingJobConfiguration doit supprimer de manière implicite tous les objets de récepteur qui ont été créés automatiquement s'ils ne sont plus utilisés par suite d'une modification de la configuration du travail d'enregistrement.

5.17 GetRecordingJobConfiguration

GetRecordingJobConfiguration doit retourner la configuration actuelle d'un travail d'enregistrement.

DEMANDE:

JobToken [tt:RecordingJobReference]
Identifie le travail d'enregistrement pour lequel extraire la configuration.

RÉPONSE:

JobConfiguration [tt:RecordingJobConfiguration}
Configuration actuelle du travail d'enregistrement.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:NoRecordingJob
JobToken ne fait pas référence à un travail existant.

CLASSE D'ACCÈS:

READ_MEDIA

5.18 SetRecordingJobMode

SetRecordingJobMode doit modifier le mode du travail d'enregistrement. L'utilisation de cette méthode doit équivaloir à extraire la configuration du travail d'enregistrement et la réécrire dans un mode différent.

Noter que l'état d'un travail d'enregistrement ne devient actif que si ce travail présente la priorité la plus élevée par rapport à tous les travaux actifs d'un enregistrement.

DEMANDE:

JobToken [tt:RecordingJobReference]
Identifie le travail d'enregistrement pour lequel modifier le mode d'enregistrement.

Mode [tt:RecordingJobMode]
Nouveau mode du travail d'enregistrement.

RÉPONSE:

Ceci est un message vide.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:NoRecordingJob
JobToken ne fait pas référence à un travail existant.

env:Sender – ter:InvalidArgVal – ter:BadMode
Le Mode n'est pas valide.

CLASSE D'ACCÈS:

ACTUATE

5.19 GetRecordingJobState

GetRecordingJobState retourne l'état d'un travail d'enregistrement. Il inclut un état agrégé et un état pour chaque piste du travail d'enregistrement. RecordingJobState peut varier du fait des

- appels qui altèrent le RecordingJobMode, par exemple, SetRecordingJobMode,
- modifications internes de l'état du moteur d'enregistrement,

- modifications du profil multimédia local enregistré, ou
- modifications de la connexion RTSP définie par le récepteur (Receiver) associé.

DEMANDE:

JobToken [tt:RecordingJobReference]
Identifie le travail d'enregistrement pour lequel obtenir l'état.

RÉPONSE:

State [tt:RecordingJobReference]
État du travail d'enregistrement.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:NoRecordingJob
JobToken ne fait pas référence à un travail existant.

CLASSE D'ACCÈS:

READ_MEDIA

La représentation UML de la structure RecordingJobStateInformation est présentée à la Figure 5.

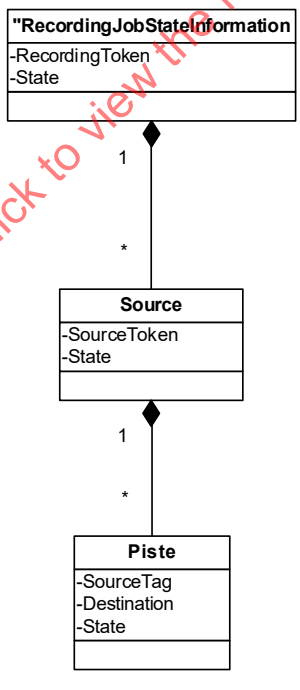

Figure 5 – Structure RecordingJobStateInformation

RecordingToken doit être l'identification de l'enregistrement dans lequel s'enregistre le travail d'enregistrement.

State (partie de RecordingJobStateInformation) doit contenir l'état agrégé sur l'ensemble de la structure RecordingJobInformation.

SourceToken doit identifier la source de données du travail d'enregistrement.

State (partie de RecordingJobStateSource) doit contenir l'état agrégé sur toutes les sous-structures de RecordingJobStateSource.

SourceTag doit identifier la piste de la source de données qui fournit les données.

Destination doit indiquer la piste de destination.

State (partie de RecordingJobTrackState) doit fournir l'état de travail de la piste. Les valeurs valides de l'état doivent être "Idle", "Active" et "Error". Si l'état est "Error" ("Erreur"), le champ Error peut être renseigné avec une valeur définie par la mise en œuvre.

Error, chaîne facultative décrivant l'état d'erreur. Il convient que la chaîne soit rédigée en anglais. Les valeurs suivantes sont prédéfinies:

"Incompatible Stream" ("Flux incompatible") – Le flux ne peut pas être enregistré parce que l'enregistrement ne correspond pas aux données enregistrées précédemment.

Un dispositif doit appliquer les règles suivantes pour déterminer l'état agrégé:

Idle	Toutes les valeurs d'état des sous-nœuds sont "Idle"
PartiallyActive	L'état de certains sous-nœuds est "Active" et celui d'autres sous-nœuds est "Idle"
Active	L'état de tous les sous-nœuds est "Active"
Error	Au moins un des sous-nœuds est à l'état "Error"

5.20 GetRecordingOptions

GetRecordingOptions retourne les informations relatives à un enregistrement identifié par le RecordingToken. Les informations comprennent le nombre de pistes supplémentaires, ainsi que de travaux d'enregistrement qui peuvent être configurés.

Cette méthode doit être prise en charge si la prise en charge de Options est indiquée par les fonctionnalités.

Noter que ces informations ne sont pas statiques et que leur validité n'est garantie que jusqu'à la prochaine modification des travaux d'enregistrement ou des pistes.

Les options de pistes doivent être prises en charge si le dispositif indique la prise en charge des pistes dynamiques.

DEMANDE:

RecordingToken [tt:RecordingReference]
Identifie l'enregistrement.

RÉPONSE:

JobOptions [trc:JobOptions]
Contient deux attributs:

Spare: Nombre de travaux de rechange qui peuvent être créés pour l'enregistrement. Par l'absence de réglage des deux attributs Spare, le dispositif indique qu'aucun travail ne peut être créé.

CompatibleSources: Un dispositif qui prend en charge l'enregistrement d'un ensemble limité de profils de service multimédia doit retourner la liste des profils qui peuvent être enregistrés sur l'enregistrement indiqué.

TrackOptions [trc:TrackOptions]
Contient quatre attributs:

SpareTotal: Nombre total de pistes de rechange qui peuvent être ajoutées à cet enregistrement.

SpareVideo: Nombre de pistes vidéo de rechange pour cet enregistrement

SpareAudio: Nombre de pistes audio de rechange pour cet enregistrement

SpareMetadata: Nombre de pistes de métadonnées de rechange pour cet enregistrement

Par l'absence de réglage des quatre attributs SpareXXX, le dispositif indique qu'aucune piste ne peut être ajoutée.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:NoRecording
RecordingToken ne fait pas référence à un enregistrement existant.

CLASSE D'ACCÈS:

READ_MEDIA

5.21 ExportRecordedData

ExportRecordedData exporte les enregistrements sélectionnés vers la cible de stockage indiquée.

Un dispositif qui indique une fonctionnalité SupportedExportFileFormats doit prendre en charge cette commande. Pour le paramètre FileFormat, il doit accepter tout format qu'il annonce par la fonctionnalité SupportedExportFileFormats.

DEMANDE:

StartPoint – facultatif [xs:dateTime]
Spécifie l'heure de début de l'exportation.

EndPoint – facultatif [xs:dateTime]
Spécifie l'heure de fin de l'exportation.

SearchScope [tt:SearchScope]
Définit le critère de sélection des enregistrements existants.

FileFormat [xs:string]
Indique quel format de fichier d'exportation utiliser.

StorageDestination [tt:StorageReferencePath]
Indique le stockage cible et le chemin d'accès au répertoire relatif.

RÉPONSE:

OperationToken [tt:ReferenceToken}
Jeton d'opération asynchrone destiné à associer l'événement reçu avec cette invocation.

FileNames – facultatif, non limité [xs:string]
Liste des noms de fichiers exportés. Le dispositif peut également utiliser l'événement AsyncronousOperationStatus pour contrôler le déroulement de l'opération ExportRecordedData.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:BadConfiguration:
Le dispositif ne peut pas prendre en charge le FileFormat sélectionné pour les fichiers exportés.

CLASSE D'ACCÈS:

READ_MEDIA

Il convient qu'un dispositif retourne la liste résultante des noms de fichiers dans la réponse. Dans les cas où l'extraction des noms de fichiers est une opération de plus longue durée, ces noms peuvent être consignés uniquement par le AsyncronousOperationStatus.

La ExportRecordedDataResponse retourne le jeton d'opération unique utilisé par le client afin de contrôler le déroulement de cette invocation. Le déroulement d'une opération d'exportation des enregistrements est obtenu par un événement (Core Spec, Monitoring/AsynchronousOperationStatus). Un dispositif peut notifier le déroulement de cette opération par l'élément FileProgressStatus (tt:ArrayOfFileProgress) facultatif, contenant un groupe de noms de fichiers et le pourcentage d'avancement du téléchargement des fichiers du point de vue du dispositif. La valeur normalisée du pourcentage d'avancement est comprise entre 0 et 1, où 1 indique un téléversement des fichiers à 100 %. L'élément FileProgressStatus facultatif est transmis dans la section Données (Data) d'un message.

Si une demande de suppression d'enregistrement est émise lors d'une exportation d'enregistrements et lorsqu'il existe des enregistrements communs, le dispositif doit supprimer l'enregistrement pertinent après l'avoir exporté.

5.22 StopExportRecordedData

Interrompt l'opération ExportRecordedData commencée auparavant. Le message de réponse répertorie le statut des fichiers exportés.

Un dispositif qui indique une fonctionnalité SupportedExportFileFormats doit prendre en charge cette commande.

DEMANDE:

OperationToken [tt:ReferenceToken]
Identifie l'opération ExportRecordedData à interrompre.

RÉPONSE:

Progress [xs:float]
Pourcentage d'avancement du téléchargement de la totalité des fichiers du point de vue du dispositif. La valeur normalisée du pourcentage d'avancement est comprise entre 0,0 et 1,0, où 1,0 indique un téléchargement de la totalité des fichiers à 100 %.

FileProgressStatus [tt:ArrayOfFileProgress]
Groupe de noms de fichiers et déroulement individuel pour chaque fichier. La valeur normalisée du pourcentage d'avancement est comprise entre 0,0 et 1,0, où 1,0 indique un téléversement des fichiers à 100 %.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
L'opération demandée n'existe pas.

CLASSE D'ACCÈS:

READ_MEDIA

5.23 GetExportRecordedDataState

GetExportRecordedDataState retourne le statut des opérations d'exportation. Cette interface permet au client d'interroger les informations de statut provenant du dispositif.

Un dispositif qui indique une fonctionnalité SupportedExportFileFormats doit prendre en charge cette commande.

DEMANDE:

OperationToken [tt:ReferenceToken]
Identifie l'opération ExportRecordedData de laquelle obtenir le statut.

RÉPONSE:

Progress [xs:float]
Pourcentage d'avancement du téléchargement de la totalité des fichiers du point de vue du dispositif. La valeur normalisée du pourcentage d'avancement est comprise entre 0,0 et 1,0, où 1,0 indique un téléversement de la totalité des fichiers à 100 %.

FileProgressStatus [tt:ArrayOfFileProgress]
Groupe de noms de fichiers et déroulement individuel pour chaque fichier. La valeur normalisée du pourcentage d'avancement est comprise entre 0,0 et 1,0, où 1,0 indique un téléversement des fichiers à 100 %.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:BadConfiguration
L'opération demandée n'existe pas.

CLASSE D'ACCÈS:

READ_MEDIA

5.24 GetServiceCapabilities

Les fonctionnalités reflètent les fonctions facultatives et la fonctionnalité d'un service. Les informations sont statiques et ne varient pas pendant le fonctionnement du dispositif. Les fonctionnalités suivantes sont disponibles:

DynamicRecordings — Indique si le dispositif prend en charge la création et la suppression dynamiques des enregistrements.

DynamicTracks — Indique si le dispositif prend en charge la création et la suppression dynamiques des pistes.

Encoding Indique quels codages sont pris en charge pour l'enregistrement. La liste peut contenir une ou plusieurs valeurs d'énumération de tt:VideoEncoding et tt:AudioEncoding. Pour les codages qui ne sont ni définis dans tt:VideoEncoding, ni dans tt:AudioEncoding, le dispositif doit utiliser les définitions IANA (voir [IANA Media Types]). Noter qu'un dispositif sans prise en charge audio ne doit pas retourner de codages audio.

MaxRate Débit binaire maximal en kBit/s, pris en charge pour toutes les pistes d'un enregistrement.

MaxTotalRate Débit binaire maximal en kBit/s, pris en charge pour tous les enregistrements.

MaxRecordings Nombre maximal d'enregistrements pris en charge.

MaxRecordingJobs Nombre maximal total de travaux d'enregistrement pris en charge par le dispositif.

Options Indique si le dispositif prend en charge la commande GetRecordingOptions.

MetadataRecording Indique si le dispositif prend en charge l'enregistrement des métadonnées.

SupportedExportFileFormats Indique que le dispositif prend en charge la commande ExportRecordedData pour les formats de fichiers d'exportation répertoriés. Un dispositif doit retourner uniquement cette fonctionnalité s'il contient au moins une valeur de format de fichiers d'exportation. La valeur de l'"ONVIF" fait référence à la spécification du format de fichiers d'exportation telle que définie dans le présent document.

DEMANDE:

Ceci est un message vide.

RÉPONSE:

Capabilities [trc:Capabilities]
Liste des fonctionnalités définies ci-dessus.

DÉFAUTS:

Aucun

CLASSE D'ACCÈS:

PRE_AUTH

5.25 Événements

5.25.1 Généralités

Certains de ces événements s'apparentent à ceux générés automatiquement et qui peuvent faire l'objet d'une recherche par la méthode FindEvents du service de recherche (voir l'Article 6).

5.25.2 Modifications d'état des travaux d'enregistrement

Si le champ d'état de la structure RecordingJobStateInformation varie, un dispositif doit générer l'événement suivant:

```
Topic: tns1:RecordingConfig/JobState
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingJobToken"
Type="tt:RecordingJobReference"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="State" Type="xs:String"/>
    <tt:ElementItemDescription Name="Information"
Type="tt:RecordingJobStateInformation"/>
  </tt:Data>
</tt:MessageDescription>
```

5.25.3 Modifications de configuration

Si la configuration d'un enregistrement est modifiée, un dispositif doit générer l'événement suivant:

```
Topic: tns1:RecordingConfig/RecordingConfiguration
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Configuration" Type="tt:RecordingConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

Si la configuration d'une piste est modifiée, un dispositif doit générer l'événement suivant:

```
Topic: tns1:RecordingConfig/TrackConfiguration
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
    <tt:SimpleItemDescription Name="TrackToken" Type="tt:TrackReference"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Configuration" Type="tt:TrackConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

Si la configuration d'un travail d'enregistrement est modifiée, un dispositif doit générer l'événement suivant:

```
Topic: tns1:RecordingConfig/RecordingJobConfiguration
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingJobToken"
Type="tt:RecordingJobReference"/>
  </tt:Source>
  <tt:Data>
    <tt:ElementItemDescription Name="Configuration"
Type="tt:RecordingJobConfiguration"/>
  </tt:Data>
</tt:MessageDescription>
```

5.25.4 Suppression de données

À chaque suppression de données, un dispositif doit générer l'événement suivant:

```
Topic: tns1:RecordingConfig/DeleteTrackData
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
    <tt:SimpleItemDescription Name="TrackToken" Type="tt:TrackReference"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="StartTime" Type="xsDateTime"/>
    <tt:SimpleItemDescription Name="EndTime" Type="xsDateTime"/>
  </tt:Data>
</tt:MessageDescription>
```

5.25.5 Enregistrement et création et suppression de pistes

À chaque création d'un enregistrement, un dispositif doit générer l'événement suivant:

```
Topic: tns1:RecordingConfig/CreateRecording
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
  </tt:Source>
  <tt:Data>
  </tt:Data>
</tt:MessageDescription>
```

À chaque suppression d'un enregistrement, un dispositif doit générer l'événement suivant:

```
Topic: tns1:RecordingConfig/DeleteRecording
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
  </tt:Source>
  <tt:Data>
  </tt:Data>
</tt:MessageDescription>
```

À chaque création d'une piste, un dispositif doit générer l'événement suivant:

```
Topic: tns1:RecordingConfig/CreateTrack
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
    <tt:SimpleItemDescription Name="TrackToken" Type="tt:TrackReference"/>
  </tt:Source>
  <tt:Data>
  </tt:Data>
</tt:MessageDescription>
```

À chaque suppression d'une piste, un dispositif doit générer l'événement suivant:

```
Topic: tns1:RecordingConfig/DeleteTrack
<tt:MessageDescription IsProperty="false">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken" Type="tt:RecordingReference"/>
    <tt:SimpleItemDescription Name="TrackToken" Type="tt:TrackReference"/>
  </tt:Source>
  <tt:Data>
  </tt:Data>
</tt:MessageDescription>
```

5.26 Exemples

5.26.1 Exemple 1: configuration de l'enregistrement d'une seule caméra

L'opération comporte deux étapes. La première consiste à configurer le NVS

```
; Create recording (this implicitly creates an A, V and M track)
RecordToken = CreateRecording(RecordConfiguration)

; The tracktokens are predefined. We don't have to find them on the
device
TrackToken1 = "VIDEO001"
TrackToken2 = "AUDIO001"
TrackToken3 = "META001"

; Create a recording job, assume that we set mode to idle, auto create
receiver
JobToken, ActualJobConfig = CreateRecordingJob(JobConfiguration)

; Configure the receiver
ConfigureReceiver(ActualJobConfiguration.ReceiverToken,
ReceiverConfiguration)
```

Ceci achève l'étape de configuration.

Enfin, pour véritablement démarrer l'enregistrement, une entité appelle

```
; Activate the recording job
SetRecordingJobMode(JobToken, Active)
```

pour activer le travail. Cette opération permet au NVS de configurer une connexion RTSP avec le dispositif.

Par conséquent, pour démarrer et interrompre l'enregistrement, la seule opération nécessaire consiste à appeler SetRecordingJobMode sur des travaux d'enregistrement préalablement configurés. Étant donné que les objets de configuration intégrés sont persistants, il est nécessaire de réaliser le cycle de configuration une fois seulement.

5.26.2 Exemple 2: enregistrement de plusieurs flux d'une caméra vers un seul enregistrement

Cet exemple est très similaire à l'exemple 1 (voir 5.26.1). La configuration du travail contient des références à deux objets de récepteur. Chaque objet de récepteur est configuré à partir du même dispositif, mais à partir d'un autre flux.

```
; Create recording (this implicitly creates an A, V and M track)
RecordToken = CreateRecording(RecordConfiguration)

; The tracktokens are predefined. We don't have to find them on the device
TrackToken1 = "VIDEO001"
TrackToken2 = "AUDIO001"
TrackToken3 = "META001"

; Create three additional tracks
TrackToken4 = CreateTrack(RecordToken, AudioConfig)
TrackToken5 = CreateTrack(RecordToken, VideoConfig)
TrackToken6 = CreateTrack(RecordToken, MetadataConfig)


; Create a recording job, assume that we set mode to idle,auto create two
receivers
JobToken, ActualJobConfiguration = CreateRecordingJob(JobConfiguration)


; Configure the receivers
ConfigureReceiver(ActualJobConfiguration.ReceiverToken[1],
                                                Receiver1Configuration)
ConfigureReceiver(ActualJobConfiguration.ReceiverToken[2],
                                                Receiver2Configuration)
```

Pour véritablement démarrer l'enregistrement, une entité appelle

```
; Activate the recording job
SetRecordingJobMode(JobToken, Active)
```

6 Service de recherche

6.1 Généralités

Le service de recherche propose un certain nombre d'opérations de recherche de données intéressantes dans un ensemble d'enregistrements. La méthode de réalisation la plus courante consiste à rechercher les événements qui sont soit inclus dans la piste de métadonnées d'un enregistrement, soit associés à un enregistrement dans le dispositif (voir 6.2.2).

GetRecordingSummary retourne un récapitulatif pour tous les enregistrements, et peut être utilisé pour fournir l'échelle d'une chronologie.

GetRecordingInformation retourne des informations relatives à un seul enregistrement (l'heure de début et le statut actuel, par exemple).

GetMediaAttributes retourne les attributs multimédias d'un enregistrement à un instant donné.

La recherche réelle a lieu sous la forme d'opérations couplées de recherche et de résultats. Chaque opération de recherche lance une session de recherche. Le client peut alors obtenir les résultats issus de la session de recherche par incréments ou en totalité, selon la mise en œuvre et l'étendue de la recherche. Il existe quatre paires d'opérations de recherche pour les enregistrements, les événements d'enregistrement, les positions PTZ et les métadonnées.

EndSearch met fin à une session de recherche, en interrompant la recherche et en retournant toutes les opérations de résultat bloquantes.

6.2 Concepts

6.2.1 Direction de recherche

Une recherche est réalisée à partir d'un point de départ de la chronologie vers un point d'extrémité. Si le point d'extrémité précède le point de départ, la recherche est réalisée en sens inverse. Cela peut être utile si seul l'événement correspondant le plus récent présente un intérêt ou s'il est commode d'obtenir les résultats du plus récent au plus ancien.

Si aucun point d'extrémité n'est précisé, la recherche est toujours réalisée à partir du point de départ.

6.2.2 Événement d'enregistrement

Décrit un événement discret lié à l'enregistrement. Il est représenté sous la forme d'un message de notification, ce qui ne signifie pas nécessairement qu'il a été enregistré en tant que notification. Les événements d'enregistrement peuvent être des notifications incluses dans une piste de métadonnées enregistrées, être créés par le dispositif d'enregistrement par suite d'un événement ou un mécanisme interne, ou être insérés par un client à l'aide d'une demande WebService ou d'un flux de métadonnées. Toutefois, l'événement d'enregistrement a été créé et associé à un enregistrement particulier, la présente spécification ne précisant pas comment il est stocké en interne sur le dispositif, mais uniquement comment il convient de le représenter dans l'interface.

Peu importe la manière dont ils ont été créés, les événements d'enregistrement sont toujours traités comme des notifications par rapport aux filtres de recherche et aux résultats retournés. Chaque événement d'enregistrement comporte une rubrique de notification telle que définie dans 10.7 de l'IEC 60839-11-31:2016. Les événements d'enregistrement prédéfinis sont décrits en 6.17.

Pour communiquer l'état d'origine des événements de propriété, des événements d'état de démarrage virtuels peuvent être retournés dans un résultat de recherche contenant la valeur d'une ou de plusieurs propriétés au point de départ de l'intervalle de recherche. Ces événements d'état de départ sont virtuels dans le sens où ils sont créés à la volée par le serveur, plutôt que collectés à partir des données enregistrées. Si le client indique que ces événements sont souhaités en définissant le drapeau approprié, les événements virtuels correspondant aux rubriques définies dans le filtre de recherche doivent être retournés pour tous les enregistrements de l'étendue de la recherche.

6.2.3 Session de recherche

Une session de recherche est démarrée de manière asynchrone par une opération de recherche et est identifiée par un jeton de recherche unique à cette session. Les résultats sont retournés par incréments dans des opérations GetResult se rapportant à la session créée par l'opération de recherche (Find). La recherche peut prendre fin de trois manières:

- Expiration du délai KeepAlive – Si aucune demande faisant référence à une session particulière n'est formulée par un client dans le temps imparti, la recherche prend fin.
- Une méthode GetResult retourne les dernières données de la session de recherche en attribuant la valeur "Completed" ("Terminé") dans le résultat de l'état de recherche.
- EndSearch – Le client met explicitement fin à une session.

La fin d'une session annule une recherche en cours, le retour immédiat et la formulation d'autres requêtes adressées à la même session donnant lieu à un message d'erreur. Un dispositif ne doit pas réutiliser immédiatement le jeton de recherche, cela risquant de troubler les clients ne sachant pas qu'une session a pris fin.

6.2.4 Étendue de la recherche

6.2.4.1 Généralités

L'étendue contient un certain nombre d'éléments facultatifs, chacun limitant l'ensemble de données à consulter lors de l'exécution de recherches.

6.2.4.2 Données incluses

Le client peut éventuellement définir les sources et les enregistrements dans lesquels procéder à la recherche en spécifiant des listes de jetons pour chaque type. En présence de plusieurs types, l'union des jetons spécifiés doit être utilisée. Si aucune source ni aucun jeton d'enregistrement ne sont spécifiés, tous les enregistrements doivent être inclus. L'étendue est affinée par le filtre d'informations d'enregistrement (Recording Information Filter). Toutefois, si des enregistrements sont spécifiés, les filtres sont uniquement appliqués à ce sous-ensemble d'enregistrements.

6.2.4.3 Filtre d'informations d'enregistrement

Plutôt que de spécifier une liste de jetons d'enregistrement, les enregistrements peuvent être filtrés par un filtre XPath fonctionnant sur la structure RecordingInformation. Cela permet au client de filtrer tous les éléments présents dans la structure RecordingInformation, en utilisant des comparaisons en fonction du dialecte XPath défini en 6.18. Si un filtre d'informations d'enregistrement est fourni, seuls les enregistrements correspondant au filtre doivent faire partie de l'étendue.

Exemple de filtre incluant uniquement des enregistrements contenant l'élément audio dans l'étendue de la recherche:

```
boolean(//Track[TrackType = "Audio"])
```

6.2.5 Filtres de recherche

Les filtres de recherche sont spécifiques au type d'opération de recherche. Voir FindEvents, FindPTZPosition et FindMetadata. Ils agissent tous sur les enregistrements définis par l'étendue.

6.2.6 Informations temporelles

Un dispositif doit prendre en charge les valeurs de temps des paramètres de demande qui sont indiquées en temps UTC avec l'indicateur 'Z', et répondre à toutes ces valeurs en tant que valeurs UTC y compris l'indicateur 'Z'.

6.3 Structures de données

6.3.1 Structure RecordingInformation

RecordingInformation contient des informations relatives à un enregistrement, aux pistes dont il est composé et à la source.

- RecordingToken – identifiant unique de l'enregistrement.
- EarliestRecording – date et heure des données les plus anciennes dans l'enregistrement.
- LatestRecording – date et heure des données les plus récentes dans l'enregistrement.
- Content – description informative du contenu. Si le contenu est inconnu, ce champ est vide.
- RecordingStatus – statut actuel de l'enregistrement, qui peut être l'un des suivants: Initiated (Lancé), Recording (Enregistrement en cours), Stopped (Interrompu), Removing (Supression en cours), Removed (Retiré), Unknown (Inconnu).

- RecordingSourceInformation – structure contenant des informations relatives à la source de l'enregistrement.
- TrackInformation – liste des structures d'informations de piste.

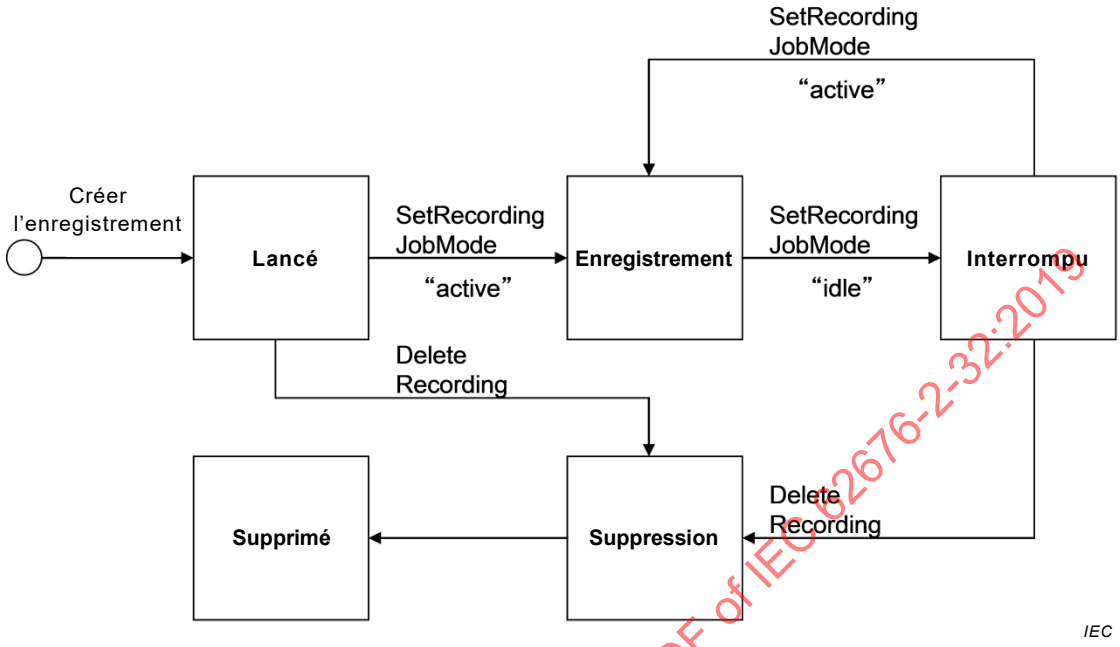

Figure 6 – Organigramme des états d'enregistrement

La Figure 6 présente les modifications d'état de RecordingStatus. Les statuts suivants apportent une explication complémentaire:

- Initiated – le nouvel enregistrement est créé et l'enregistrement spécifié n'a pas encore démarré.
- Recording – le travail d'enregistrement procède à l'enregistrement spécifié.
- Stopped – le travail d'enregistrement interrompt provisoirement l'enregistrement spécifié. Ce cas exige que le travail d'enregistrement ait déjà commencé au moins une fois.
- Removing – l'enregistrement spécifié est en cours de suppression.
- Removed – l'enregistrement spécifié a été supprimé.
- Unknown – il convient que ce cas ne se produise jamais.

6.3.2 Structure RecordingSourceInformation

Contient des informations relatives à la source d'un enregistrement.

- SourceId – identifiant de la source choisie par le client qui crée l'enregistrement. Cet identifiant est opaque pour le dispositif. Les clients peuvent utiliser tout type d'URI pour ce champ. En l'absence de cet identifiant, ce champ est vide.
- Name – nom informatif de la source. Si le nom est inconnu, ce champ est vide.
- Location – description informative de l'emplacement de la source. Si la description n'est pas disponible, cette chaîne est vide.
- Description – description informative de la source. En l'absence de cette description, la description est vide.
- Address – URI informatif de la source.

6.3.3 Structure TrackInformation

Contient des informations relatives à une seule piste dans un enregistrement. Noter qu'une piste peut représenter un intervalle de temps contigu unique ou être composée de plusieurs tranches comme cela est présenté en 5.1. En l'absence de données enregistrées pour une piste, TrackInformation ne doit pas être fournie.

- TrackToken – identifiant de la piste. TrackToken est unique parmi tous ceux utilisés dans un enregistrement.
- TrackType – identifie le type de piste (vidéo, audio ou métadonnées)
- Description – description informative de la piste. En l'absence de cette information, ce champ est vide.
- DataFrom – date et heure de début d'enregistrement des données les plus anciennes de la piste.
- DataTo – date et heure d'interruption d'enregistrement des données les plus récentes de la piste.

6.3.4 Énumération SearchState

L'état de la recherche peut être l'un des suivants

- Searching (Recherche en cours) – La base de données est en cours de recherche et certains résultats peuvent être disponibles et être extraits au moyen de la méthode GetEventSearchResults.
- Completed (Terminée) – La recherche est terminée et tous les résultats ont été fournis avec GetEventSearchResults.

L'utilisation de l'état défini complémentaire "Queued" ("En file d'attente") est déconseillée.

6.3.5 Structure MediaAttributes

L'élément MediaAttributes contient des informations relatives aux pistes multimédias d'un enregistrement particulier pour un bloc de temps particulier. Le bloc de temps peut être un point dans le temps, auquel cas les éléments From et Until sont identiques.

- RecordingToken – référence à l'enregistrement que cette structure concerne.
- From – point dans le temps à partir duquel les attributs sont valides pour l'enregistrement.
- Until – point dans le temps auquel les attributs spécifiés cessent d'être valides pour l'enregistrement.
- VideoAttributes – ensemble d'attributs vidéo, décrivant les données d'une piste vidéo enregistrée.
- AudioAttributes – ensemble d'attributs audio, décrivant les données d'une piste audio enregistrée.
- MetadataAttributes – ensemble d'attributs, décrivant le contenu possible de métadonnées d'une piste de métadonnées enregistrée.

6.3.6 Structure FindEventResult

- RecordingToken – identifiant de l'enregistrement contenant l'événement trouvé.
- TrackToken – identifiant de la piste contenant l'événement trouvé.
- Time – date et heure de l'événement trouvé.
- Event – message d'événement trouvé.
- StartStateEvent – si la valeur est True, indique que l'événement représente l'état d'origine d'une ou de plusieurs propriétés de l'enregistrement.

6.3.7 Structure FindPTZPositionResult

- RecordingToken – identifiant de l'enregistrement contenant la position correspondante.
- TrackToken – identifiant de la piste contenant la position correspondante.
- Time – date et heure de la position correspondante.
- Position – vecteur PTZ correspondant.

6.3.8 Structure PTZPositionFilter

Contient les éléments nécessaires à la définition des positions PTZ à rechercher. Les vecteurs PTZ doivent se situer dans le même espace de coordonnées que les coordonnées PTZ stockées dans l'enregistrement.

- MinPosition – limite inférieure du volume PTZ à rechercher.
- MaxPosition – limite supérieure du volume PTZ à rechercher.
- EnterOrExit – si la valeur est True, consigner les positions lors de l'entrée ou de la sortie du volume PTZ spécifié. À défaut, consigner toutes les positions enregistrées dans la plage PTZ spécifiée.

6.3.9 Structure MetadataFilter

Contient une expression XPath à appliquer à la structure MetadataStream.

Exemple d'expression de recherche d'objets chevauchant le quadrant inférieur droit de la scène:

```
boolean(//Object/Appearance/Shape/BoundingBox[@right  >  ”0.5”])
and
boolean(//Object/Appearance/Shape/BoundingBox[@bottom > ”0.5”])
```

6.3.10 Structure FindMetadataResult

- RecordingToken – identifiant de l'enregistrement contenant les métadonnées correspondantes.
- TrackToken – identifiant de la piste contenant les métadonnées correspondantes.
- Time – date et heure des métadonnées correspondantes.

6.4 GetRecordingSummary

GetRecordingSummary permet d'obtenir une description récapitulative de toutes les données enregistrées. Cette opération est obligatoire pour la prise en charge d'un dispositif mettant en œuvre le service de recherche d'enregistrement. Si un dispositif retourne NumberRecordings comme valeur nulle, les deux éléments DataFrom et DataUntil peuvent être ignorés en toute sécurité.

DEMANDE:

Ceci est un message vide.

RÉPONSE:

Summary [tt:RecordingSummary]
Structure contenant: DataFrom spécifiant la première fois que les données ont été enregistrées sur le dispositif, DataUntil spécifiant la dernière fois que les données ont été enregistrées sur le dispositif, ainsi que le nombre total d'enregistrements sur le dispositif.

DÉFAUTS:

Aucun

CLASSE D'ACCÈS:

READ_MEDIA

6.5 GetRecordingInformation

Retourne des informations relatives à un seul Enregistrement spécifié par un RecordingToken. Cette opération est obligatoire pour la prise en charge d'un dispositif mettant en œuvre le service de recherche d'enregistrement.

DEMANDE:

RecordingToken [tt:ReferenceToken]
Identifie l'enregistrement.

RÉPONSE:

RecordingInformation [tt:RecordingInformation]
Informations relatives à l'enregistrement.

DÉFAUTS:

- **env:Sender – ter:InvalidArgVal – ter:InvalidToken**

Le jeton d'enregistrement n'est pas valide.

CLASSE D'ACCÈS:

READ_MEDIA

6.6 GetMediaAttributes

Retourne un ensemble d'attributs multimédia pour toutes les pistes des enregistrements spécifiés à un instant donné. Les clients utilisant cette opération doivent pouvoir l'utiliser comme une opération non bloquante. Un dispositif doit attribuer des valeurs égales aux heures de début et de fin de la structure MediaAttributes si le calcul de cette plage provoque le blocage de cette opération. Voir la structure MediaAttributes pour de plus amples informations. Cette opération est obligatoire pour la prise en charge d'un dispositif mettant en œuvre le service de recherche d'enregistrement.

Les dispositifs qui indiquent CanContainPTZ doivent consigner les espaces PTZ utilisés à un instant donné (y compris les espaces génériques). Pour une interopérabilité optimale, il convient que les mises en œuvre du dispositif utilisent des noms génériques[2]. Les espaces génériques sont les suivants (voir 8.9.2 de l'IEEE 62676-2-31:2019):

http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace

http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace

2 Un espace PTZ est identifié par une URI. L'URI n'est utilisée que pour créer un identifiant unique, elle ne sert à extraire aucun document.

DEMANDE:

RecordingTokens – facultatif, non limité [tt:ReferenceToken]
Liste de références aux enregistrements à demander. Si aucun jeton d'enregistrement n'est fourni, il convient de demander tous les enregistrements.

Time [xs:dateTime]
Instant de demande des informations.

RÉPONSE:

MediaAttributes – facultatif, non limité [tt:MediaAttributes]
Contient une structure MediaAttributes pour le RecordingToken spécifié dans la demande. Noter que chaque RecordingToken peut produire aucun ou un élément MediaAttributes.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:InvalidToken
Le jeton d'enregistrement n'est pas valide.

CLASSE D'ACCÈS:

READ_MEDIA

6.7 FindRecordings

FindRecordings démarre une session de recherche des enregistrements correspondant à l'étendue (voir 6.2.4) définie dans la demande. Les résultats de la recherche sont obtenus à l'aide de la demande GetRecordingSearchResults, spécifiant le jeton de recherche retourné de cette demande.

Le dispositif doit continuer la recherche jusqu'à ce que l'un des événements suivants se produise:

- le nombre total de correspondances a été trouvé, défini par le paramètre MaxMatches;
- une demande EndSearch du client a mis fin à la session;
- la session a pris fin, car KeepAliveTime depuis la dernière demande liée à cette session a expiré.

L'ordre des résultats n'est pas défini, pour permettre au dispositif de retourner les résultats dans l'ordre dans lequel il les trouve. Cette opération est obligatoire pour la prise en charge d'un dispositif mettant en œuvre le service de recherche d'enregistrement.

Pour l'option KeepAliveTime, un dispositif doit prendre en charge des valeurs au moins jusqu'à 10 s. Un dispositif peut adapter des valeurs plus grandes.

DEMANDE:

Scope [tt:SearchScope]
Définit le jeu de données à prendre en compte pour cette recherche.

MaxMatches – facultatif [xs:int]
La recherche se termine après MaxMatches.

KeppAliveTimeout [xs:duration]
Délai d'attente de la session après chaque demande concernant cette session.

RÉPONSE:

SearchToken [tt:Jobtoken]
Identifie la session de recherche créée par cette demande.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:InvalidToken
Le jeton d'enregistrement n'est pas valide.

env:Sender – ter:InvalidArgVal – ter:InvalidSource
La source d'enregistrement n'est pas valide.

env:Receiver – ter:Action – ter:ResourceProblem
Le dispositif n'est pas capable de créer une nouvelle session de recherche.

CLASSE D'ACCÈS:

READ_MEDIA

6.8 GetRecordingSearchResults

GetRecordingSearchResults obtient les résultats provenant d'une session de recherche d'enregistrement déjà initiée par une opération FindRecordings. La réponse ne doit pas contenir de résultats déjà retournés dans des demandes précédentes de la même session. Si MaxResults est spécifié, la réponse ne doit pas contenir plus de résultats que MaxResults. Le nombre de résultats est lié au nombre d'enregistrements. Utiliser la méthode FindEvents pour visualiser les données enregistrées individuelles propres à une piste de signal.

GetRecordingSearchResults doit bloquer tant que:

- les résultats MaxResults ne sont pas disponibles pour la réponse, si MaxResults est spécifié;
- les résultats MinResults ne sont pas disponibles pour la réponse, si MinResults est spécifié;
- WaitTime n'a pas expiré;
- la recherche n'est pas terminée ou interrompue.

Cette opération est obligatoire pour la prise en charge d'un dispositif mettant en œuvre le service de recherche d'enregistrement. Si les paramètres spécifiés MinResults et WaitTime dépassent la plage prise en charge, un dispositif doit les adapter et non répondre par une erreur.

DEMANDE:

SearchToken [tt:JobToken]
Spécifie la session de recherche.

MinResults – facultatif [xs:int]
Spécifie le nombre minimal de résultats qu'il convient de retourner. Si le nombre total de résultats est inférieur à MinResults dans une recherche terminée, il convient de retourner tous les résultats.

MaxResults – facultatif [xs:int]
Spécifie le nombre maximal de résultats à retourner.

WaitTime – facultatif [xs:duration]
Définit la durée maximale du blocage, en attente des résultats.

RÉPONSE:

ResultList [tt:FindRecordingResultList]
Structure contenant le SearchState actuel et une liste de structures RecordingInformation.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:InvalidToken
Le jeton de recherche n'est pas valide.

CLASSE D'ACCÈS:

READ_MEDIA

6.9 FindEvents

FindEvents démarre une session de recherche d'événements dans l'étendue (voir 6.2.4) correspondant au filtre de recherche défini dans la demande. Les événements sont des événements d'enregistrement (voir 6.2.2) et d'autres événements disponibles dans la piste. Les résultats de la recherche sont obtenus à l'aide de la demande GetEventSearchResults, spécifiant le jeton de recherche retourné de cette demande.

Le dispositif doit continuer la recherche jusqu'à ce que l'un des événements suivants se produise:

- la recherche a porté sur toute la plage de temps comprise entre StartPoint et EndPoint;
- le nombre total de correspondances a été trouvé, défini par le paramètre MaxMatches;
- une demande EndSearch du client a mis fin à la session;
- la session a pris fin, car KeepAliveTime depuis la dernière demande liée à cette session a expiré.

Les résultats doivent être ordonnés de manière chronologique, dans l'ordre croissant dans le cas d'une recherche vers l'avant ou dans l'ordre décroissant dans le cas d'une recherche vers l'arrière. Cette opération est obligatoire pour la prise en charge d'un dispositif mettant en œuvre le service de recherche d'enregistrement. Bien que les valeurs des événements de propriété fassent référence à un sens de recherche vers l'avant, elles doivent être consignées de manière identique dans le mode de recherche inversée.

Pour l'option KeepAliveTime, un dispositif doit prendre en charge des valeurs au moins jusqu'à 10 s. Un dispositif peut adapter des valeurs plus grandes.

DEMANDE:

StartPoint [xs:dateTime]
Point de départ dans le temps de la recherche.

EndPoint – facultatif [xs:dateTime]
Point d'interruption dans le temps de la recherche. Il peut s'agir d'une heure qui précède StartPoint, auquel cas la recherche est réalisée à rebours dans le temps. Si EndPoint est ignoré, la recherche est réalisée vers l'avant à partir de StartPoint.

Scope [tt:SearchScope]
Définit le jeu de données à prendre en compte pour cette recherche.

SearchFilter [tt:EventFilter]
Contient le filtre de rubrique et de message nécessaire à la définition des événements à rechercher.

IncludeStartState [xs:boolean]
Si le client attribue la valeur "true" à IncludeStartState, il indique qu'il convient de retourner des événements virtuels au moment de StartPoint pour représenter l'état dans l'enregistrement. Dans le cas d'une recherche vers l'arrière, il convient de retourner des événements virtuels au moment de StartPoint et de EndPoint. La prise en charge des événements virtuels est obligatoire pour les événements d'enregistrement. La prise en charge des événements virtuels supplémentaires est indiquée au moyen de la fonctionnalité GeneralStartEvents.

MaxMatches – facultatif [xs:int]
La recherche se termine après MaxMatches.

KeepAliveTime [xs:duration]
Délai d'attente de la session après chaque demande concernant cette session.

RÉPONSE:

SearchToken [tt:JobToken]
Identifie la session de recherche créée par cette demande.

DÉFAUTS:

env:Receiver – ter:Action – ter:ResourceProblem
Le dispositif n'est pas capable de créer une nouvelle session de recherche.

env:Sender – ter:InvalidArgVal – ter:InvalidFilterFault
L'expression de filtre de recherche fournie n'a pas été comprise ou prise en charge par le dispositif.

CLASSE D'ACCÈS:

READ_MEDIA

6.10 GetEventSearchResults

GetEventSearchResults obtient les résultats provenant d'une session de recherche d'événement d'enregistrement déjà initiée par une opération FindEvents. La réponse ne doit pas contenir de résultats déjà retournés dans des demandes précédentes de la même session. Si MaxResults est spécifié, la réponse ne doit pas contenir plus de résultats que MaxResults.

GetEventSearchResults doit bloquer tant que:

- les résultats MaxResults ne sont pas disponibles pour la réponse, si MaxResults est spécifié;
- les résultats MinResults ne sont pas disponibles pour la réponse, si MinResults est spécifié;
- WaitTime n'a pas expiré;
- la recherche n'est pas terminée ou interrompue.

Cette opération est obligatoire pour la prise en charge d'un dispositif mettant en œuvre le service de recherche d'enregistrement. Si les paramètres spécifiés MinResults et WaitTime dépassent la plage prise en charge, un dispositif doit les adapter et non répondre par une erreur.

DEMANDE:

SearchToken [tt:JobToken]
Spécifie la session de recherche.

MinResults – facultatif [xs:int]
Spécifie le nombre minimal de résultats qu'il convient de retourner. Si le nombre total d'événements restants est inférieur à MinResults dans une recherche terminée, il convient de retourner tous ces événements.

MaxResults – facultatif [xs:int]
Spécifie le nombre maximal de résultats à retourner.

WaitTime – facultatif [xs:duration]
Définit la durée maximale du blocage, en attente des résultats.

RÉPONSE:

ResultList [tt: FindEventResultList]
Structure contenant:

L'état de la recherche lorsque le résultat est retourné. Indique s'il peut y avoir plus de résultats, ou si la recherche est terminée.

Structure FindEventResult pour chaque événement trouvé correspondant à la recherche.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:InvalidToken
Le jeton de recherche n'est pas valide.

CLASSE D'ACCÈS:

READ_MEDIA

6.11 FindPTZPosition

FindPTZPosition démarre une session de recherche positions PTZ dans l'étendue (voir 6.2.4) correspondant au filtre de recherche défini dans la demande. Les résultats de la recherche sont obtenus à l'aide de la demande GetPTZPositionSearchResults, spécifiant le jeton de recherche retourné de cette demande.

Le dispositif doit continuer la recherche jusqu'à ce que l'un des événements suivants se produise:

- la recherche a porté sur toute la plage de temps comprise entre StartPoint et EndPoint;
- le nombre total de correspondances a été trouvé, défini par le paramètre MaxMatches;
- une demande EndSearch du client a mis fin à la session;
- la session a pris fin, car KeepAliveTime depuis la dernière demande liée à cette session a expiré.

Il est obligatoire de prendre en charge cette opération à chaque fois que la valeur de CanContainPTZ est "true" pour toutes les pistes de métadonnées d'un enregistrement du dispositif.

Pour l'option KeepAliveTime, un dispositif doit prendre en charge des valeurs au moins jusqu'à 10 s. Un dispositif peut adapter des valeurs plus grandes.

Un dispositif doit uniquement correspondre aux critères de recherche par rapport aux mises à jour du statut PTZ disponibles entre l'intervalle de temps indiqué dans la recherche, c'est-à-dire que le dispositif ne doit pas localiser la position PTZ au début de l'intervalle de recherche.

DEMANDE:

StartPoint [xs:dateTime]
Point de départ dans le temps de la recherche.

EndPoint – facultatif [xs:dateTime]
Point d'interruption dans le temps de la recherche. Il peut s'agir d'une heure qui précède StartPoint, auquel cas la recherche est réalisée à rebours dans le temps. Si EndPoint est ignoré, la recherche est réalisée vers l'avant à partir de StartPoint.

Scope [tt:SearchScope]
Définit le jeu de données à prendre en compte pour cette recherche.

SearchFilter [tt: PTZPositionFilter]
Contient les critères de recherche nécessaires à la définition de la position PTZ à rechercher.

MaxMatches – facultatif [xs:int]
La recherche se termine après MaxMatches.

KeepAliveTime [xs:duration]
Délai d'attente de la session après chaque demande concernant cette session.

RÉPONSE:

SearchToken [tt:JobToken]
Identifie la session de recherche créée par cette demande.

DÉFAUTS:

env:Receiver – ter:Action – ter:ResourceProblem
Le dispositif n'est pas capable de créer une nouvelle session de recherche.

env:Receiver – ter:ActionNotSupported – ter:NotImplemented
Cette méthode facultative n'est pas mise en œuvre

CLASSE D'ACCÈS:

READ_MEDIA

6.12 GetPTZPositionSearchResults

GetPTZPositionSearchResults obtient les résultats provenant d'une session de recherche de position PTZ déjà initiée par une opération FindPTZPosition. La réponse ne doit pas contenir de résultats déjà retournés dans des demandes précédentes de la même session. Si MaxResults est spécifié, la réponse ne doit pas contenir plus de résultats que MaxResults.

GetPTZPositionSearchResults doit bloquer tant que:

- les résultats MaxResults ne sont pas disponibles pour la réponse, si MaxResults est spécifié;
- les résultats MinResults ne sont pas disponibles pour la réponse, si MinResults est spécifié;
- WaitTime n'a pas expiré;
- la recherche n'est pas terminée ou interrompue.

Il est obligatoire de prendre en charge cette opération à chaque fois que la valeur de CanContainPTZ est "true" pour toutes les pistes de métadonnées d'un enregistrement du dispositif. Si les paramètres spécifiés MinResults et WaitTime dépassent la plage prise en charge, un dispositif doit les adapter et non répondre par une erreur.

DEMANDE:

SearchToken [tt:JobToken]
Spécifie la session de recherche.

MinResults – facultatif [xs:int]
Spécifie le nombre minimal de résultats qu'il convient de retourner. Si le nombre total d'événements restants est inférieur à MinResults dans une recherche terminée, il convient de retourner tous ces événements.

MaxResults – facultatif [xs:int]
Spécifie le nombre maximal de résultats à retourner.

WaitTime – facultatif [xs:duration]
Définit la durée maximale du blocage, en attente des résultats.

RÉPONSE:

ResultList [tt:FindPTZPositionResultList]
Structure contenant:

L'état de la recherche lorsque le résultat est retourné. Indique s'il peut y avoir plus de résultats, ou si la recherche est terminée.

Structure FindPTZPositionResult pour chaque position PTZ trouvée correspondant à la recherche.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:InvalidToken
Le jeton de recherche n'est pas valide.

CLASSE D'ACCÈS:

READ_MEDIA

6.13 FindMetadata

FindMetadata démarre une session de recherche de métadonnées dans l'étendue (voir 6.2.4) correspondant au filtre de recherche défini dans la demande. Les résultats de la recherche sont obtenus à l'aide de la demande GetMetadataSearchResults, spécifiant le jeton de recherche retourné de cette demande.

Le dispositif doit continuer la recherche jusqu'à ce que l'un des événements suivants se produise:

- la recherche a porté sur toute la plage de temps comprise entre StartPoint et EndPoint;
- le nombre total de correspondances a été trouvé, défini par le paramètre MaxMatches;
- une demande EndSearch du client a mis fin à la session;
- la session a pris fin, car KeepAliveTime depuis la dernière demande liée à cette session a expiré.

Il est obligatoire de prendre en charge cette opération si la valeur "true" est attribuée à la fonctionnalité MetaDataSearch dans la structure SearchCapabilities retournée par la commande GetCapabilities dans le service de dispositif.

Pour l'option KeepAliveTime, un dispositif doit prendre en charge des valeurs au moins jusqu'à 10 s. Un dispositif peut adapter des valeurs plus grandes.

DEMANDE:

StartPoint [xs:dateTime]
Point de départ dans le temps de la recherche.

EndPoint – facultatif [xs:dateTime]
Point d'interruption dans le temps de la recherche. Il peut s'agir d'une heure qui précède StartPoint, auquel cas la recherche est réalisée à rebours dans le temps. Si EndPoint est ignoré, la recherche est réalisée vers l'avant à partir de StartPoint.

Scope [tt:SearchScope]
Définit le jeu de données à prendre en compte pour cette recherche.

SearchFilter [tt: MetadataFilter]
Contient les critères de recherche nécessaires à la définition des métadonnées à rechercher.

MaxMatches – facultatif [xs:int]
La recherche se termine après MaxMatches.

KeepAliveTime [xs:duration]
Délai d'attente de la session après chaque demande concernant cette session.

RÉPONSE:

SearchToken [tt:JobToken]
Identifie la session de recherche créée par cette demande.

DÉFAUTS:

env:Receiver – ter:Action – ter:ResourceProblem
Le dispositif n'est pas capable de créer une nouvelle session de recherche.

CLASSE D'ACCÈS:

READ_MEDIA

6.14 GetMetadataSearchResults

GetMetadataSearchResults obtient les résultats provenant d'une session de recherche d'enregistrement déjà initiée par une opération FindMetadata. La réponse ne doit pas contenir de résultats déjà retournés dans des demandes précédentes de la même session. Si MaxResults est spécifié, la réponse ne doit pas contenir plus de résultats que MaxResults.

GetMetadataSearchResults doit bloquer tant que:

- les résultats MaxResults ne sont pas disponibles pour la réponse, si MaxResults est spécifié;
- les résultats MinResults ne sont pas disponibles pour la réponse, si MinResults est spécifié;
- WaitTime n'a pas expiré;
- la recherche n'est pas terminée ou interrompue.

Il est obligatoire de prendre en charge cette opération si la valeur "true" est attribuée à la fonctionnalité MetaDataSearch dans la structure SearchCapabilities retournée par la commande GetCapabilities dans le service de dispositif. Si les paramètres spécifiés MinResults et WaitTime dépassent la plage prise en charge, un dispositif doit les adapter et non répondre par une erreur.

DEMANDE:

SearchToken [tt:JobToken]
Spécifie la session de recherche.

MinResults – facultatif [xs:int]
Spécifie le nombre minimal de résultats qu'il convient de retourner. Si le nombre total d'événements restants est inférieur à MinResults dans une recherche terminée, il convient de retourner tous ces événements.

MaxResults – facultatif [xs:int]
Spécifie le nombre maximal de résultats à retourner.

WaitTime – facultatif [xs:duration]
Définit la durée maximale du blocage, en attente des résultats.

RÉPONSE:

ResultList [tt: FindMetadataResultList]
Structure contenant:

L'état de la recherche lorsque le résultat est retourné. Indique s'il peut y avoir plus de résultats, ou si la recherche est terminée.

Structure FindMetadataResult pour chaque jeu de métadonnées trouvé correspondant à la recherche.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:InvalidToken
Le jeton de recherche n'est pas valide.

CLASSE D'ACCÈS:

READ_MEDIA

6.15 EndSearch

EndSearch interrompt une session de recherche en cours, provoquant le retour de toutes les demandes de résultats bloquantes et l'invalidité de SearchToken. Si la recherche a été interrompue avant la fin, le point dans le temps que la recherche a atteint doit être retourné. Si la recherche n'a pas encore commencé, StartPoint doit être retourné. Noter qu'un message d'erreur apparaît si la session de recherche est déjà terminée avant cette demande. Si la recherche est terminée, le EndPoint d'origine fourni par l'opération Find doit être retourné. À l'émission d'un message EndSearch lors d'une demande FindRecordings, le EndPoint n'est pas défini et ne doit pas être utilisé étant donné que cette demande ne comporte pas de StartPoint/EndPoint. Cette opération est obligatoire pour la prise en charge d'un dispositif mettant en œuvre le service de recherche d'enregistrement.

DEMANDE:

SearchToken [tt:Jobtoken]
Spécifie la session de recherche.

RÉPONSE:

EndPoint [xs:dateTime]
Point dans le temps auquel la recherche se trouvait lors de l'interruption.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:InvalidToken
Le jeton de recherche n'est pas valide.

CLASSE D'ACCÈS:

READ_MEDIA

6.16 GetServiceCapabilities

Les fonctionnalités reflètent les fonctions facultatives et la fonctionnalité d'un service. Les informations sont statiques et ne varient pas pendant le fonctionnement du dispositif. Les fonctionnalités suivantes sont disponibles:

MetadataSearch Indique si le dispositif prend en charge la recherche générique des métadonnées enregistrées

GeneralStartEvents Indique la prise en charge des événements de propriété virtuels généraux dans la méthode FindEvents

DEMANDE:

Ceci est un message vide.

RÉPONSE:

Capabilities [tse:Capabilities]
Contient les fonctionnalités de service demandées en utilisant une structure de fonctionnalité XML hiérarchique.

CLASSE D'ACCÈS:

PRE_AUTH

6.17 Descriptions d'événements d'enregistrement

Un dispositif doit générer les événements suivants avec les descriptions de message d'événement correspondantes. Un dispositif qui prend en charge le service de recherche d'enregistrement doit enregistrer ces messages de notification de sorte que les clients puissent utiliser FindEvents pour les rechercher. Tous les événements d'enregistrement générés par le dispositif et insérés dans l'historique des enregistrements doivent comporter une rubrique racine de tns1:RecordingHistory.

L'attribut UtcTime du NotficationMessage (voir 10.5.3 de l'IEC 60839-11-31:2016) associé à tous les événements d'enregistrement doit spécifier la durée réelle relative à l'enregistrement indépendamment du temps d'envoi de l'événement.

```
Topic: tns1:RecordingHistory/Recording/State
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken"
Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="IsRecording" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

Ce message est envoyé à chaque fois qu'un client démarre ou interrompt un enregistrement spécifique. Au démarrage de l'enregistrement, la valeur "true" doit être attribuée à IsRecording. À l'interruption de l'enregistrement, la valeur "false" doit être attribuée à IsRecording.

```
Topic: tns1:RecordingHistory/Track/State
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="RecordingToken"
Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescription Name="Track" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="IsDataPresent" Type="xs:boolean"/>
  </tt:Data>
</tt:MessageDescription>
```

Ce message indique la présence de données pour une piste. Lorsque les données deviennent présentes, un message dont la valeur True est attribuée à IsDataPresent doit être envoyé. Lorsque les données deviennent indisponibles, le message dont la valeur False est attribuée à IsDataPresent doit être envoyé.

Un dispositif peut générer les événements suivants. Si le dispositif prend en charge ces événements, il doit toujours enregistrer automatiquement ces messages de notification de sorte que les clients puissent toujours utiliser FindEvent pour ces messages.

Topic: tns1:RecordingHistory/Track/VideoParameters

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="Recording"
Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescription Name="Track" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="VideoEncoding"
              Type="tt:VideoEncoding"/>
    <tt:SimpleItemDescription Name="VideoWidth" Type="xs:int"/>
    <tt:SimpleItemDescription Name="VideoHeight" Type="xs:int"/>
    <tt:ElementItemDescription                    Name="VideoRateControl"
              Type="tt:VideoRateControl"/>
  </tt:Data>
</tt:MessageDescription>
```

Topic: tns1:RecordingHistory/Track/AudioParameters

```
<tt:MessageDescription IsProperty="true">
  <tt:Source>
    <tt:SimpleItemDescription Name="Recording"
Type="tt:ReferenceToken"/>
    <tt:SimpleItemDescription Name="Track" Type="tt:ReferenceToken"/>
  </tt:Source>
  <tt:Data>
    <tt:SimpleItemDescription Name="AudioEncoding"
              Type="tt:AudioEncoding"/>
    <tt:SimpleItemDescription Name="AudioSampleRate" Type="xs:int"/>
    <tt:SimpleItemDescription Name="AudioBitrate" Type="xs:int"/>
  </tt:Data>
</tt:MessageDescription>
```

Le dispositif doit envoyer un des deux messages (selon le type de données de piste) à chaque modification de ces propriétés.

6.18 Dialecte XPath

Le présent paragraphe définit un dialecte W3C XPATH 1.0 qu'un dispositif qui exécute le service de recherche doit mettre en œuvre pour procéder à l'analyse syntaxique des chaînes Xpath transmises aux méthodes du service de recherche.

```
Dialect=http://www.onvif.org/ver10/tse/searchFilter

[1] Expression::= BoolExpr | Expression 'and' Expression
     | Expression 'or' Expression | '('Expression')' |
'not''('Expression')'
[2] BoolExpr::= 'boolean''('PathExpr')' | 'contains''(' ElementPath ',' '"'
String '"' ')'
[3] PathExpr::= '//SimpleItem' NodeTest | '//ElementItem' NodeTest |
     ElementTest
[4] NodeTest::= '['AttrExpr']'
[5] AttrExpr::= NameComp | ValueComp | AttrExpr 'and' AttrExpr | AttrExpr
'or' AttrExpr | 'not''(AttrExpr')'
[6] NameComp::= NameAttr'='"'String'"'
[7] ValueComp::= ValueAttr Operator '"'String'"'
[8] Operator::= '=' | '!=' | '<' | '<=' | '>' | '>='
[9] NameAttr::= '@Name'
[10] ValueAttr::= '@Value'
[11] ElementTest::= '/' ElementPath '['NodeComp']'
[12] ElementPath::= ElementName ElementName*
[13] ElementName::= '/' String
[14] NodeComp::= NodeName Operator '"' String '"'
[15] NodeName::= '@' String | String
```

Exemple d'expression XPath utilisée pour rechercher des enregistrements provenant de la base et contenant au moins une piste vidéo:

```
boolean(//Source[Location = “Basement”]) and
boolean(//Track[TrackType = “Video”])
```

7 Contrôle de lecture

7.1 Demande d'URI de lecture

GetReplayUri demande un URI qui peut être utilisé pour initier la restitution d'un flux enregistré en utilisant le protocole RTSP en tant que protocole de commande. L'URI est valide uniquement tel qu'il est spécifié dans la réponse. Toutes les mises en œuvre du service de lecture doivent prendre en charge la commande GetReplayUri.

DEMANDE:

StreamSetup [tt:StreamSetup]
L'élément StreamSetup contient deux parties. StreamType définit si un flux multimédia à monodiffusion ou multidiffusion est demandé. Transport spécifie une chaîne de protocoles de transport définissant la tunnellisation du flux multimédia par différents protocoles réseau.

RecordingToken [tt:RefernceToken]
Indique l'enregistrement à soumettre à un flux.

RÉPONSE:

Uri [xs:anyURI]
Contient l'URI à utiliser dans le cadre de la demande de flux multimédia.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:InvalidStreamSetup
La spécification de la partie StreamType ou Transport dans StreamSetup n'est pas prise en charge.

env:Sender – ter:OperationProhibited – ter:StreamConflict
La spécification de la partie StreamType ou Transport dans StreamSetup génère un conflit avec d'autres flux.

env:Sender – ter:InvalidArgVal – ter:NoRecording
L'enregistrement n'existe pas.

CLASSE D'ACCÈS:

READ_MEDIA

7.2 ReplayConfiguration

La ReplayConfiguration fournit des informations de configuration facultatives. Actuellement, elle ne contient que la SessionTimeout RTSP déconseillée qui peut être contrôlée par la couche RTSP.

7.3 SetReplayConfiguration

SetReplayConfiguration modifie la configuration du service de lecture. Le service de lecture doit permettre de modifier sa configuration à l'aide de cette commande.

DEMANDE:

Configuration [tt:ReplayConfiguration]
Contient la nouvelle configuration du service de lecture.

RÉPONSE:

Ceci est un message vide.

DÉFAUTS:

env:Sender – ter:InvalidArgVal – ter:ConfigModify
Les valeurs de la configuration ne peuvent pas être définies.

CLASSE D'ACCÈS:

ACTUATE

7.4 GetReplayConfiguration

GetReplayConfiguration retourne la configuration actuelle du service de lecture. Le service de lecture doit permettre d'extraire sa configuration à l'aide de cette commande.

DEMANDE:

Ceci est un message vide.

RÉPONSE:

Configuration [tt:ReplayConfiguration]
Contient la configuration actuelle du service de lecture.

DÉFAUTS:

Aucun

CLASSE D'ACCÈS:

READ_MEDIA

7.5 GetServiceCapabilities

Les fonctionnalités reflètent les fonctions facultatives et la fonctionnalité d'un service. Les informations sont statiques et ne varient pas pendant le fonctionnement du dispositif. Les fonctionnalités suivantes sont disponibles:

ReversePlayback Indique que le dispositif prend en charge la restitution inversée telle que définie en 8.6.

SessionTimeoutRange Répertorie les limites supérieure et inférieure de la plage prise en charge pour le délai d'attente de la session. Cette fonctionnalité indique par défaut la valeur RTSP par défaut.

RTP_RTSP_TCP Indique si le dispositif prend en charge le transport RTP/RTSP/TCP, voir 10.2.1.4 de l'IEC 62676-2-31:2019.

RTSPOverWebSocket Indique si le dispositif prend en charge la transmission en continu de restitution RTSP/RTP par WebSocket et fournit l'URI WebSocket de lecture comme cela est décrit en 10.2.1.6 de l'IEC 62676-2-31:2019.

DEMANDE:

Ceci est un message vide.

RÉPONSE:

Capabilities [trp:Capabilities]
Contient les fonctionnalités de service demandées en utilisant une structure de fonctionnalité XML hiérarchique.

DÉFAUTS:

Aucun

CLASSE D'ACCÈS:

PRE_AUTH

8 Restitution

8.1 Utilisation du protocole RTSP

Le protocole de lecture repose sur le protocole RTSP (IETF RFC 2326). Toutefois, étant donné que le protocole RTSP ne prend pas directement en charge nombre des caractéristiques exigées par les applications CCTV, le présent document définit plusieurs extensions vers le protocole. Ces extensions sont décrites en détail ci-dessous.

Le présent document formule les préconisations suivantes quant à l'utilisation du protocole RTSP:

1) Le serveur doit prendre en charge le protocole RTP/RTSP/HTTP/TCP. Il s'agit du même protocole de transport qu'un dispositif mettant en œuvre la transmission en continu multimédia au moyen du service multimédia doit prendre en charge, et les mêmes exigences doivent s'appliquer pour lire la transmission en continu.
2) Le serveur doit prendre en charge le transport RTP/UDP à monodiffusion pour la transmission en continu.
3) Il convient que les clients utilisent un transport TCP pour la lecture, afin d'obtenir une livraison fiable des paquets multimédias.
4) Le serveur peut choisir de ne pas envoyer de paquets RTCP lors de la lecture. Dans le cadre d'une utilisation classique, les paquets RTCP ne sont pas exigés, car en règle générale, un transport fiable est utilisé et des informations temporelles absolues sont envoyées dans le flux, rendant ces informations redondantes dans les rapports expéditeurs RTCP.

8.2 Description RTSP

Le protocole SDP retourné par la commande de description RTSP doit contenir la TrackReference de chaque piste de l'enregistrement, afin de permettre à un client de mettre en correspondance les pistes présentées dans le protocole SDP avec celles de l'enregistrement. La balise doit utiliser le format suivant:

a:x-onvif-track:<TrackReference>

Par exemple:

```
NVS – NVT:     DESCRIBE rtsp://192.168.0.1 RTSP/1.0
               Cseq: 1
               User-Agent: ONVIF Rtsp client
               Accept: application/sdp

NVT – NVS:     RTSP/1.0 200 OK
               Cseq: 1
               Content-Type: application/sdp
               Content-Length: xxx

v=0
               o= 2890842807 IN IP4 192.168.0.1
                    m=video 0 RTP/AVP 26
                    a=control:rtsp://192.168.0.1/video
               a=x-onvif-track:VIDEO001
               m=audio 0 RTP/AVP 98
               a=control:rtsp://192.168.0.1/audio
               a=x-onvif-track:AUDIO001
```

8.3 Extension d'en-tête RTP

8.3.1 Généralités

Afin de permettre aux clients de consigner un horodatage stable et exact pour chaque trame restituée, quel que soit le sens de restitution, il est nécessaire d'associer un horodatage absolu à chaque paquet ou chaque groupe de paquets au même horodatage RTP (une trame vidéo, par exemple). Pour ce faire, une extension d'en-tête RTP est utilisée, contenant un horodatage NTP et des informations supplémentaires également utiles pour la lecture.

Le mécanisme de lecture utilise l'ID d'extension 0xABAC pour l'extension de lecture.

Le Tableau 3 présente la forme générale d'un paquet RTP contenant cette extension.

Tableau 3 – Présentation d'un paquet RTP

<table>
<tr><td colspan="2">V=2</td><td>P</td><td>X=1</td><td>CC</td><td>M</td><td>PT</td><td colspan="2">numéro de séquence</td></tr>
<tr><td colspan="9">horodatage</td></tr>
<tr><td colspan="9">identifiant de source de synchronisation (SSRC)</td></tr>
<tr><td colspan="6">0xABAC</td><td colspan="3">longueur=3</td></tr>
<tr><td colspan="9">Horodatage NTP...</td></tr>
<tr><td colspan="9">...horodatage NTP</td></tr>
<tr><td>C</td><td>E</td><td>D</td><td colspan="2">mbz</td><td colspan="3">Cseq</td><td colspan="2">remplissage</td></tr>
<tr><td colspan="9">charge utile...</td></tr>
</table>

Les champs de cette extension sont les suivants:

– Horodatage NTP. Horodatage NTP (IETF RFC 1305) indiquant le temps UTC absolu associé à l'unité d'accès.

– C: 1 bit. Indique que cette unité d'accès est un point de synchronisation ou un "point propre" (le début d'une trame intracodée dans le cas de flux vidéo, par exemple);

– E: 1 bit. Indique la fin d'une section contiguë d'enregistrements. La dernière unité d'accès de chaque piste avant un écart d'enregistrement ou à la fin du métrage disponible doit comporter ce jeu de bits. En cas de lecture inversée, le drapeau E doit être défini sur la dernière trame, à la fin de la section contiguë de l'enregistrement;

– D: 1 bit. Indique que cette unité d'accès suit une discontinuité dans la transmission. Elle est principalement utilisée lors de la lecture inversée. Le bit D du premier paquet de chaque GOP est défini, puisqu'il ne suit pas chronologiquement le paquet précédent dans le flux de données (voir 8.6).

– mbz: Ce champ est réservé à une utilisation ultérieure et doit être nul.

– Cseq: 1 octet. Il s'agit de l'octet d'ordre inférieur de la valeur CSeq utilisée dans la commande RTSP PLAY qui a permis d'initier la transmission. Lorsqu'un client envoie plusieurs commandes PLAY consécutives, cette valeur peut être utilisée pour déterminer le point où commencent les données issues de chaque nouvelle commande PLAY.

L'extension d'en-tête de lecture doit être présente dans le premier paquet de chaque unité d'accès (une trame vidéo, par exemple).

8.3.2 Horodatages NTP

Les horodatages NTP de l'en-tête d'extension RTP doivent augmenter de manière monotone sur les paquets successifs à l'intérieur d'un seul flux RTP. Il convient qu'ils correspondent à l'heure pendulaire mesurée au niveau de l'émetteur d'origine du flux, ajustée si nécessaire afin de préserver le caractère monotone.

8.3.3 Compatibilité avec l'extension d'en-tête JPEG

L'extension d'en-tête de lecture peut coexister avec l'extension d'en-tête utilisée par le profil JPEG RTP, ce qui est nécessaire pour assurer la lecture des flux JPEG utilisant cette extension. L'extension JPEG est simplement ajoutée à l'extension de lecture. Sa présence est indiquée par un champ de longueur d'extension d'en-tête RTP contenant une valeur supérieure à 3, et par le code de début d'extension 0xFFD8 ou 0xFFFF au début du quatrième mot du contenu de l'extension.

Le Tableau 4 présente un paquet JPEG utilisant les deux extensions:

Tableau 4 – Présentation de paquet RTP avec en-tête JPEG

<table>
<tr><td colspan="2">V=2</td><td>P</td><td>X=1</td><td>CC</td><td>M</td><td>PT</td><td colspan="2">numéro de séquence</td></tr>
<tr><td colspan="9">horodatage</td></tr>
<tr><td colspan="9">identifiant de source de synchronisation (SSRC)</td></tr>
<tr><td colspan="6">0xABAC</td><td colspan="3">longueur=N+4</td></tr>
<tr><td colspan="9">Horodatage NTP...</td></tr>
<tr><td colspan="9">...horodatage NTP</td></tr>
<tr><td>C</td><td>E</td><td>D</td><td colspan="2">mbz</td><td colspan="2">CSeq</td><td colspan="2">remplissage</td></tr>
<tr><td colspan="6">0xFFD8</td><td colspan="3">jpeglength=N</td></tr>
<tr><td colspan="9">charge utile d'extension: séquence de segments de marqueur JPEG supplémentaires remplis de 0xFF jusqu'à la longueur d'extension totale</td></tr>
<tr><td colspan="9">charge utile</td></tr>
</table>

8.4 Balise de caractéristique RTSP

Le service de lecture utilise la balise de caractéristique "onvif-replay" pour indiquer qu'il prend en charge les extensions RTSP décrites dans le présent document.

Exemple:

```
C->S:   SETUP rtsp://server.com/foo/bar/baz.rm RTSP/1.0
        Cseq: 302
        Require: onvif-replay

S->C:   RTSP/1.0 551 Option not supported
        Cseq: 302
        Unsupported: onvif-replay
```

Le serveur de lecture doit accepter une commande SETUP et PLAY dont un en-tête Require contient la balise de caractéristique onvif-replay.

8.5 Lancement de la restitution

8.5.1 Généralités

La restitution est lancée au moyen de la méthode RTSP PLAY. Par exemple:

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
Cseq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-
Rate-Control: no
```

La fonctionnalité ReversePlayback, définie en 7.5, indique si un dispositif prend en charge la restitution inversée. La restitution inversée est indiquée par le champ d'en-tête Scale contenant une valeur négative. Par exemple, pour une lecture inversée sans perte de données, une valeur −1,0 est utilisée.

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
Cseq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-
Rate-Control: no
Scale: -1,0
```

Si un dispositif prend en charge la restitution inversée, il doit accepter un en-tête Scale contenant une valeur −1,0. Un dispositif peut accepter d'autres valeurs pour le paramètre Scale. Sauf si la valeur "no" est attribuée à l'en-tête Rate-Control (voir ci-dessous), le paramètre Scale est utilisé de la manière décrite dans la norme IETF RFC 2326. Si la valeur "no" est attribuée à Rate-Control, la valeur 1,0 ou –1,0 doit être attribuée au paramètre Scale, s'il est présent, afin d'indiquer respectivement la restitution vers l'avant ou la restitution inversée. Si ce paramètre est absent, la restitution est par hypothèse une restitution vers l'avant.

8.5.2 Champ d'en-tête Range

Un dispositif doit prendre en charge le champ Range (Plage) exprimé en utilisant des heures absolues telles que définies par la norme IETF RFC 2326. Les heures absolues sont exprimées à l'aide d'utc-range issu de la norme IETF RFC 2326.

Des plages ouvertes ou fermées peuvent être utilisées. Dans le cas d'une plage fermée, la plage augmente (heure de fin ultérieure à l'heure de début) pour la restitution vers l'avant, et diminue pour la restitution inversée. La direction de la plage doit correspondre à la valeur de l'en-tête Scale.

Dans tous les cas, le premier point de la plage indique le point de départ de la lecture.

L'heure proprement dite doit être indiquée sous la forme

```
utc-range = "clock" ["=" utc-range-spec]
utc-range-spec = ( utc-time "-" [ utc-time ] ) / ( "-" utc-time )
utc-time = utc-date "T" utc-clock "Z"
utc-date = 8DIGIT
utc-clock = 6DIGIT [ "." 1*9DIGIT ]
```

comme cela est défini dans la norme IETF RFC 2326.

Exemples:

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
Cseq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-20090615T115000Z
Rate-Control: no

PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
Cseq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T115000.440Z-20090615T114900Z
Rate-Control: no
Scale: -1,0
```

8.5.3 Champ d'en-tête Rate-Control

La présente spécification introduit le champ d'en-tête Rate-Control, auquel peut être attribuée une valeur "yes" ("oui") ou "no" ("non"). En l'absence du champ, la valeur utilisée est par hypothèse la valeur "yes", et le flux est délivré en temps réel à l'aide des mécanismes de synchronisation RTP normalisés. Si la valeur "no" est attribuée à ce champ, le flux est délivré aussi rapidement que possible, en utilisant uniquement le contrôle de flux fourni par le transport pour limiter le débit de livraison.

La différence importante entre ces deux modes est qu'avec l'expression "Rate-Control=yes", le serveur contrôle la vitesse de restitution, alors qu'avec l'expression "Rate-Control=no", c'est le client qui la contrôle.

Lors de la lecture de plusieurs pistes d'un seul enregistrement, lancée par une seule commande RTSP PLAY et n'utilisant pas le contrôle de débit, il convient de multiplexer dans le temps les données issues des pistes dans le même ordre que celui dans lequel elles ont été enregistrées.

Un dispositif doit prendre en charge l'opération avec "Rate-Control=no" pour la restitution.

8.5.4 Champ d'en-tête Frames

Le champ d'en-tête Frames (Trames) peut être utilisé pour réduire le nombre d'images (trames) transmises (pour limiter la bande passante ou la charge de traitement, par exemple). Trois modes sont possibles:

1) Images I uniquement. Indiqué par la valeur "intra", éventuellement suivie d'un intervalle minimal entre des images I successives dans le flux. Cette dernière peut être utilisée pour limiter le nombre d'images reçues, même en présence de "rafales d'images I" générées par de nombreux récepteurs demandant des images I fréquentes.
2) Images I et images P uniquement. Indiqué par la valeur "predicted" ("prédite"). Cette valeur peut être utilisée pour éliminer les images B si le flux en contient.
3) Toutes les images. Valeur par défaut.

Exemples:

Pour demander des images I uniquement:

```
Frames: intra
```

Pour demander des images I avec un intervalle minimal de 4 000 millisecondes:

```
Frames: intra/4000
```

Pour demander des images I et des images P uniquement:

```
Frames: predicted
```

Pour demander toutes les images (noter qu'il n'est pas nécessaire de spécifier explicitement ce mode, mais l'exemple est inclus par souci d'exhaustivité):

```
Frames: all
```

L'argument d'intervalle utilisé avec l'option "intra" fait référence à la chronologie d'enregistrement, pas à la durée de restitution. Par conséquent, pour un intervalle donné, les mêmes images sont diffusées, quelle que soit la vitesse de restitution. L'argument d'intervalle ne doit pas être présent, sauf si l'option Frames est "intra".

Le serveur doit prendre en charge le champ d'en-tête Frames. Il ne s'agit pas d'empêcher l'utilisation du champ d'en-tête Scale comme autre moyen de limiter le débit de données. La mise en œuvre du champ d'en-tête Scale peut varier entre les différentes mises en œuvre de serveur, comme le précise la norme IETF RFC 2326.

Un dispositif doit prendre en charge les paramètres Frames ("intra" et "all") pour la restitution.

8.5.5 Points de synchronisation

Le flux vidéo transmis doit commencer au niveau d'un point de synchronisation, voir 6.8 de l'IEC 62676-11-31:2019. Les règles de choix de la trame de début sont les suivantes:

– Si l'heure de début demandée se situe dans la section du métrage enregistré, le flux commence par le premier point propre à l'heure de début demandée ou avant. Tel est le cas, quel que soit le sens de restitution.
– Si l'heure de début demandée se situe dans un écart du métrage enregistré et que la restitution est lancée dans la direction avant, le flux commence par le premier point propre de la section suivant l'heure de début demandée.
– Si l'heure de début demandée se situe dans un écart du métrage enregistré et que la restitution est lancée dans la direction inversée, le flux commence par le dernier point propre de la section précédant l'heure de début demandée.

8.6 Lecture inversée

8.6.1 Lancement

La lecture inversée est lancée par le champ d'en-tête Scale contenant une valeur négative comme cela est décrit ci-dessus.

8.6.2 Ordre de transmission de paquets

L'exemple de la Figure 7 présente le mode de transmission des trames lors d'une restitution vers l'avant normale pour un enregistrement comportant deux vidéoclips de courte durée. Chaque vidéoclip est composé de deux GOP. Comme le représente la Figure 7, tous les paquets sont transmis dans l'ordre d'enregistrement. La dernière trame précédant un écart est marquée avec le drapeau E afin d'indiquer un écart au niveau de l'enregistrement.

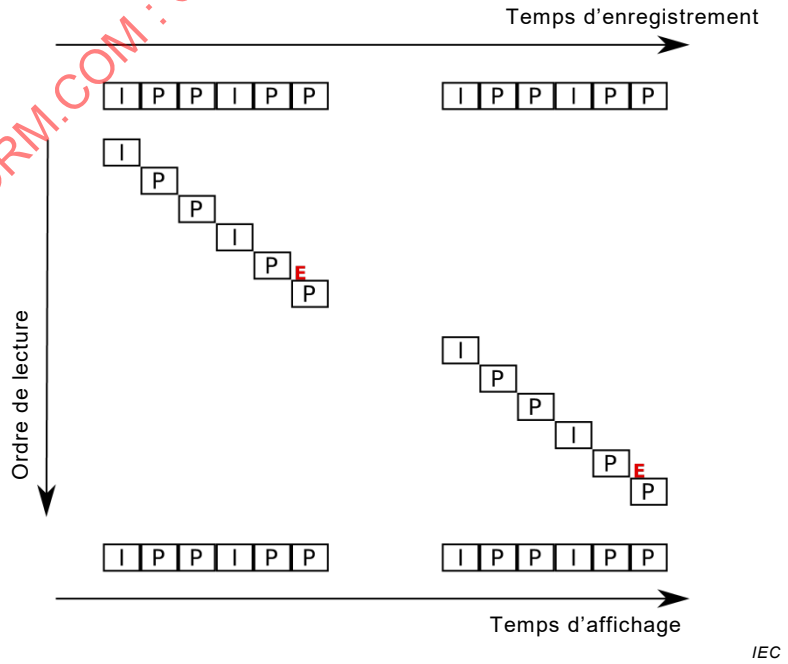

Figure 7 – Transmission de paquets lors d'une restitution vers l'avant

L'ordre de transmission des paquets vidéo lors d'une lecture inversée repose sur les GOP, un GOP étant composé d'un point propre suivi d'une séquence de paquets de points non propres.

Lors de la restitution inversée, les GOP doivent être envoyés dans l'ordre inversé, mais les paquets que contient un GOP doivent être envoyés dans l'ordre avant. L'en-tête d'extension RTP du premier paquet de chaque GOP doit comporter un jeu de bits de "discontinuité". L'en-tête d'extension RTP de la dernière trame d'un GOP suivant immédiatement un écart (ou le début du métrage disponible) doit comporter un jeu de bits E.

Lors de la transmission de trames clés uniquement, ou lorsque le codec ne repose pas sur le mouvement (JPEG, par exemple), un GOP est considéré comme étant composé d'une seule trame, mais peut toujours être composé de plusieurs paquets. Dans ce cas, les paquets de chaque trame doivent de nouveau être envoyés dans l'ordre avant, alors que les trames elles-mêmes doivent être envoyées dans l'ordre inversé.

Les flux audio et de métadonnées peuvent être transmis dans un ordre correspondant à celui du flux vidéo. Par conséquent, les paquets provenant de ces flux sont envoyés dans l'ordre de restitution vers l'avant jusqu'à l'occurrence d'un paquet (en général un paquet vidéo) contenant le jeu de bits D défini dans l'en-tête d'extension, au niveau duquel ils reviennent à un point avant la discontinuité.

Noter que la restitution inversée d'un paquet audio n'est pas utile. Par conséquent, il convient généralement de ne pas transmettre les paquets audio lors de la restitution inversée.

L'exemple de la Figure 8 présente le mode de transmission des paquets lors de la restitution vers l'avant inversée, pour le même codage que celui représenté à la Figure 7. Comme le représente la Figure 8, tous les paquets sont transmis dans l'ordre d'enregistrement.

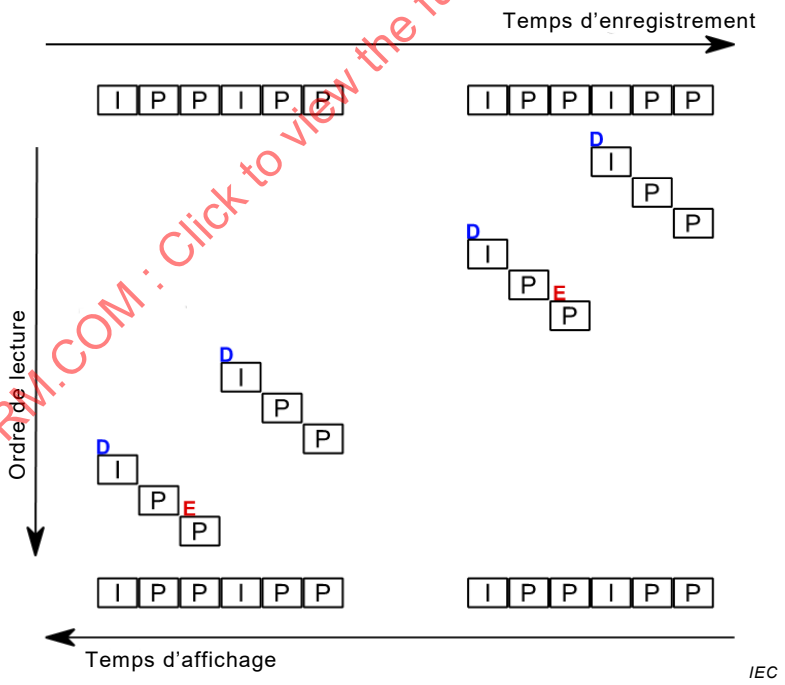

Figure 8 – Transmission de paquets lors d'une restitution inversée

8.6.3 Numéros de séquence RTP

Les numéros de séquence RTP des paquets transmis lors d'une restitution inversée doivent être incrémentés de manière monotone dans l'ordre de livraison, pas dans l'ordre de restitution prévu.

8.6.4 Horodatages RTP

L'utilisation des horodatages RTP dépend de la valeur de l'en-tête Rate-Control. Si la valeur de cet en-tête est "no" ("non") (c'est-à-dire que le client contrôle la vitesse de restitution), les horodatages RTP sont déduits des temps d'échantillonnage d'origine des trames enregistrées. En l'absence de l'en-tête Rate-Control ou si sa valeur est "yes" ("oui") (c'est-à-dire que le serveur contrôle la vitesse de restitution), les horodatages RTP correspondent à la durée de restitution telle que décrite à l'Annexe B de la norme (IETF RFC 2326).

Si la valeur de Rate-Control est "no", les horodatages RTP des paquets transmis lors de la restitution inversée doivent être identiques à ceux qui seraient utilisés si ces mêmes paquets étaient transmis dans la direction avant. À l'inverse des numéros de séquence, les horodatages RTP correspondent à l'ordre d'enregistrement d'origine, pas à l'ordre de livraison. Le serveur peut utiliser les mêmes horodatages RTP que ceux reçus à l'origine lors de l'enregistrement du flux.

Cela signifie que des paquets RTP successifs d'un seul GOP présentent toujours des horodatages RTP progressifs (voir l'ordre de transmission ci-dessus), mais que l'horodatage sur les trames d'index des GOP reçus successivement diminue lors de la lecture inversée.

Si la valeur de Rate-Control est "yes", les horodatages RTP des paquets transmis lors de la restitution inversée doivent indiquer les heures auxquelles il convient que chaque trame soit rendue au niveau du client. Ainsi, les paquets successifs d'un seul GOP présentent des horodatages RTP dégressifs (étant donné qu'il convient que le premier soit diffusé en dernier) et que les horodatages des trames d'index augmentent. Dans ce mode, l'intervalle entre des horodatages successifs dépend des valeurs des en-têtes Speed et Scale, comme cela est décrit à l'Annexe B de la norme IETF RFC 2326.

8.7 RTSP Keepalive

Lorsque le contrôle de débit est désactivé et que le flux RTP est tunnelisé par le biais de la connexion RTSP (c'est-à-dire par les transports RTP/RTSP/TCP ou RTP/RTSP/HTTP/TCP), le client doit être informé du fait qu'il peut ne pas être en mesure de recevoir la réponse à une demande, par exemple en cas de pause du processus de lecture.

8.8 Enregistrement du métrage en cours

Si le client commence la restitution à partir d'une heure réelle ou peu de temps avant, il peut finir par diffuser le métrage en temps réel au fur et à mesure de l'enregistrement. Dans ce cas, le serveur continue simplement à envoyer les données de flux au client au fur et à mesure de leur réception.

Noter que le bit E n'est pas défini sur les unités d'accès en cours d'enregistrement, même si chaque unité d'accès envoyée au client de lecture est en général la dernière connue du serveur. Toutefois, si l'enregistrement s'interrompt, le bit E est défini sur la dernière unité d'accès de l'enregistrement.

8.9 Fin de métrage

Si la restitution atteint un point au-delà duquel plus aucune donnée n'est envoyée dans un ou plusieurs flux, elle interrompt la transmission de données, mais ne passe pas à l'état "paused" ("pause"). Si le serveur reprend l'enregistrement après la survenue de cette situation, la livraison reprend au fur et à mesure de la réception des nouvelles données.

8.10 Go To Time

Comme cela est indiqué en 10.5 de la norme IETF RFC 2326, une commande PLAY reçue lorsqu'une lecture est déjà en cours n'a aucun impact tant que l'opération de lecture n'est pas terminée. La présente spécification ajoute un nouvel en-tête RTSP "Immediate", ("Immédiat") qui remplace ce comportement de la commande PLAY utilisé avec:

```
PLAY rtsp://192.168.0.1/path/to/recording RTSP/1.0
CSeq: 123
Session: 12345678
Require: onvif-replay
Range: clock=20090615T114900.440Z-
Rate-Control: no
Immediate: yes
```

Si le serveur reçoit une commande PLAY dont la valeur "yes" est attribuée à l'en-tête Immediate, il démarre immédiatement la lecture à partir du nouvel emplacement, annulant toutes les commandes PLAY existantes. L'en-tête d'extension RTP du premier paquet envoyé depuis le nouvel emplacement doit comporter le jeu de bits D (discontinuité).

Un dispositif doit prendre en charge le champ d'en-tête Immediate pour une restitution avec la valeur "yes". La spécification ne définit pas le comportement sans en-tête Immediate ou sans valeur "no".

8.11 Utilisation du protocole RTCP

Un serveur n'est pas tenu d'envoyer des paquets RTCP. S'il les envoie, les règles suivantes s'appliquent:

- Si le contrôle de débit est activé (voir 8.5.3), les paquets RTCP doivent être construits et transmis comme cela est spécifié dans la norme IETF RFC 3550. En particulier, l'horodatage NTP d'un rapport expéditeur indique l'heure pendulaire actuelle et n'est pas lié aux horodatages NTP intégrés aux en-têtes d'extension RTP des flux de données.
- Si le contrôle de débit n'est pas activé, l'horodatage NTP et l'horodatage RTP de chaque rapport expéditeur doivent être nuls.

9 Format de fichiers d'exportation

9.1 Informations collatérales nécessaires

SurveillanceExportBox est exigé. Il est recommandé de placer SurveillanceExportBox dans les fichiers le plus tôt possible, pour une utilité maximale.

Pour pouvoir associer l'enregistrement avec une caméra/un microphone et le système d'exportation, la case doit contenir les informations suivantes:

- Source – Description de la source vidéo
  - Nom (Name) – Nom de la caméra
  - URL – Adresse d'accès à la caméra
  - MAC – Adresse physique unique de la caméra (exemples: 08-00-27-00-0C-15, 08:00:27:00:0C:15, 080027000C15)
  - Ligne (Line) – Jeton de nombre de lignes d'entrée pour des dispositifs multivoies
- Source – Description de la source audio
  - Nom – Nom du microphone
  - URL – Adresse d'accès au microphone
  - MAC – Adresse physique unique du microphone
- Exportation (Export) – Unité exécutant l'exportation
  - Nom – Nom de l'unité d'exportation
  - URL – Adresse d'accès à l'unité d'exportation
  - MAC – Adresse physique unique de l'unité d'exportation

– Heure (Time) – Informations relatives à la date et à l'heure d'exécution de l'exportation (heure de début)
– Opérateur (Operator) – Nom ou identification de l'opérateur effectuant l'exportation

SurveillanceExportBox

Type de case: 'suep'
Conteneur: Case de métadonnées ('meta'), intégrée aux fichiers
Obligatoire: Oui
Quantité: Exactement une

Cette case doit fournir des informations ordonnées relatives à la source et aux unités d'exportation. Les pistes multimédias sont identifiées par leur UUID respectif. Les informations doivent être disponibles uniquement pour le type de supports présents, et être fournies séparément pour toutes les pistes. Les informations relatives à l'unité d'exportation sont les informations principales.

Syntaxe

```
class SurveillanceExportBox

      extends      FullBox('suep', version = 0, 0){
      string       ExportUnitName;
      string       ExportUnitURL;
      string       ExportUnitMAC;
      UInt(64)     ExportUnitTime;
      string       ExportOperator

      Uint(8)      video_count;
     int i;
     for (i=0; i < video_count; i++) {

            UInt(128)       VideoTrackUUID
            string          VideoSourceName;
            string          VideoSourceURL;
            string          VideoSourceMAC;
            string          VideoSourceLine;
      }

      Uint(8)  audio_count;
      int j;

     for (j=0; j < audio_count; j++) {

            UInt(128)       AudioTrackUUID
            string          AudioSourceName;
            string          AudioSourceURL;
            string          AudioSourceMAC;
      }
}
```

Sémantique

Les éléments de chaînes sont des chaînes terminées par zéro composées de caractères UTF-8. Lorsqu'elles ne sont pas applicables, la chaîne doit contenir uniquement la terminaison par zéro.