ONVIF™
ONVIF Recording Control Service Specification

Version 26.06

June, 2026

CONTENTS

1 Scope **5**

2 Normative references **5**

3 Terms and Definitions **5**

3.1 Definitions 5

4 Overview **6**

4.1 Storage 6
4.1.1 Storage Model 6
4.1.2 Recording 7
4.1.3 External targets 8

5 Recording control **8**

5.1 Introduction 8

5.2 General Requirements 9

5.3 Data structures 9
5.3.1 RecordingConfiguration 9
5.3.2 TrackConfiguration 10
5.3.3 RecordingJobConfiguration 10
5.3.4 Event recording 11

5.4 CreateRecording 12

5.5 DeleteRecording 12

5.6 GetRecordings 13

5.7 SetRecordingConfiguration 13

5.8 GetRecordingConfiguration 14

5.9 CreateTrack 14

5.10 DeleteTrack 15

5.11 GetTrackConfiguration 16

5.12 SetTrackConfiguration 16

5.13 CreateRecordingJob 17

5.14 DeleteRecordingJob 17

5.15 GetRecordingJobs 18

5.16 SetRecordingJobConfiguration 18

5.17 GetRecordingJobConfiguration 19

5.18 SetRecordingJobMode 19

5.19 GetRecordingJobState 20

5.20 GetRecordingOptions 21

5.21 ExportRecordedData 22

5.22 StopExportRecordedData 23

5.23 GetExportRecordedDataState 23

5.24 OverrideSegmentDuration 24

5.25 GetServiceCapabilities 24

5.26 ListRecordedSegments 26
5.27 ExportRecordedSegments 26
5.28 StopExportRecordedSegments 27
5.29 Events 28
5.29.1 Recording job state changes 28
5.29.2 Configuration changes 28
5.29.3 Data deletion 29
5.29.4 Recording and track creation and deletion 29
5.30 Examples 30
5.30.1 Example 1: setup recording of a single camera 30
5.30.2 Example 2: Record multiple streams from one camera to a single recording 30
Annex A Example scenario for Recording Job Priority (Informative) 32
Annex B Object storage recording format (Normative) 33
B.1 Overview 33
B.2 Spans 33
B.3 Segments 34
B.4 Segment objects 34
B.5 Span end objects 35
B.6 Recording format MP4 35
B.7 Recording format CMAF 35
B.8 Encryption 36
B.8.1 W3C Common Encryption System 36
B.8.1.1 Syntax 36
B.8.1.2 Semantics 36
B.8.2 Asymmetric key system 37
B.8.2.1 Syntax 37
B.8.2.2 Semantics 37
B.8.2.3 Encryption Version 1 38
B.8.2.4 Encryption Version 2 38
B.9 Object attributes 39
B.9.1 Timescale Attribute 40
B.9.2 SeekTable Attribute 40
B.10 Examples 41
Annex C Segment export (Normative) 43
C.1 Overview 43
C.2 Recording segments 43
C.3 Segment export commands 43
C.4 Segment stability 44
Annex D Revision History 45

1 Scope

This document defines the web service interface for the configuration of recording of Video, Audio and Metadata. Additionally associated events are defined.

The overview section provides a definition of the ONVIF storage model. This is common for all ONVIF storage related services.

Web service usage is outside of the scope of this document. Please refer to the ONVIF core specification.

2 Normative references

IANA Algorithm registry for Hybrid Public Key Encryption (HPKE) <https://www.iana.org/assignments/hpke/hpke.xhtml>
ISO/IEC 14496-12:2022 — Information technology — Coding of audio-visual objects — Part 12: ISO base media file format <https://www.iso.org/standard/83102.html>
ISO/IEC 14496-14:2020 — Information technology — Coding of audio-visual objects — Part 14: MP4 file format <https://www.iso.org/standard/79110.html>
ISO/IEC 23000-19:2020 — Information technology — Multimedia application format (MPEG-A) — Part 19: Common media application format (CMAF) for segmented media <https://www.iso.org/standard/79106.html>
ISO/IEC 23001-7:2016 — Information technology — MPEG systems technologies — Part 7: Common encryption in ISO base media file format files <https://www.iso.org/standard/68042.html>
ISO 8601-1:2019 — Date and time — Representations for information interchange — Part 1: Basic rules <https://www.iso.org/standard/70907.html>
RFC 5234 — Augmented BNF for Syntax Specifications: ABNF <https://www.ietf.org/rfc/rfc5234.txt>
RFC 6381 — The 'Codecs' and 'Profiles' Parameters for "Bucket" Media Types <https://www.ietf.org/rfc/rfc6381.txt>
RFC 9180 — Hybrid Public Key Encryption (HPKE) <https://www.rfc-editor.org/rfc/rfc9180.html>
ONVIF Core Specification <http://www.onvif.org/specs/core/ONVIF-Core-Specification.pdf>
ONVIF Schedule Service Specification <https://www.onvif.org/specs/srv/sched/ONVIF-Scheduler-Service-Spec.pdf>
W3C "cenc" Initialization Data Format <https://www.w3.org/TR/2016/NOTE-eme-initdata-cenc-20160915/>

3 Terms and Definitions

3.1 Definitions

Metadata	All streaming data except video and audio, including video analytics results, PTZ position data and other metadata (such as textual data from POS applications).
Recording	A container for a set of audio, video and metadata tracks. A recording can hold one or more tracks. A track is viewed as an infinite timeline that holds data at certain times.
Recording Event	An event associated with a Recording, represented by a notification message in the APIs
Recording Job	A job performs the transfer of data from a data source to a particular recording using a particular configuration
Track	An individual data channel consisting of video, audio, or metadata. This definition is consistent with the definition of track in [RFC 2326]
Video Analytics	Algorithms or programs used to analyze video data and to generate data describing object location and behaviour.

4 Overview

4.1 Storage

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

The following functions are supported:

- Recording Control
- Search
- Replay

These functions are provided by three interrelated services:

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.

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,

Replay 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.

WSDL for this service is specified in http://www.onvif.org/onvif/ver10/recording.wsdl.

Table 1: Referenced namespaces (with prefix)

Prefix	Namespace URI
env	http://www.w3.org/2003/05/soap-envelope
ter	http://www.onvif.org/ver10/error
xs	http://www.w3.org/2001/XMLSchema
tt	http://www.onvif.org/ver10/schema
trc	http://www.onvif.org/ver10/recording/wsdl

4.1.1 Storage Model

The storage interfaces in this standard 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 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 e.g. 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 may 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.

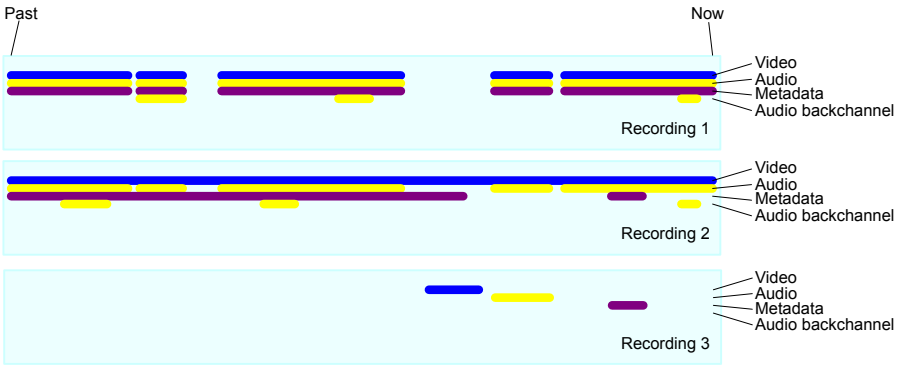

Figure 1: Storage Model with Tracks

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 standard 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 the chapter on event handling and the MetadataConfiguration object). In addition, a device also conceptually records ONVIF defined historical events (see Recording Event Descriptions in the search service), this includes information like 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.

4.1.2 Recording

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.

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.

If a device supports scheduled recording, clients may configure scheduled recording by adding a scheduler token to the recording job. A recording job with a scheduler token will only record when the associated schedule is active. If the associated schedule of a recording job is inactive a job with lower recording priority may record.

By default a recording job is continuously recording. Devices supporting the EventFilter can be configured such that they only record events. To provide some context time intervals before and after the event may be captured.

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

4.1.3 External targets

The target interface allows configuration for devices that support recording to external storage targets. For authentication configuration see the related storage configuration APIs of the core specification.

The target format structure defines for each recording a path on the storage target where the related recordings should be stored. In order to keep reasonably sized files the recordings are split into segments. This specification defines a date and time based index. Indexing according to events and video content is outside of the scope of this specification.

An encryption configuration interface allows to encrypt the content according to well defined standards.

5 Recording control

5.1 Introduction

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.

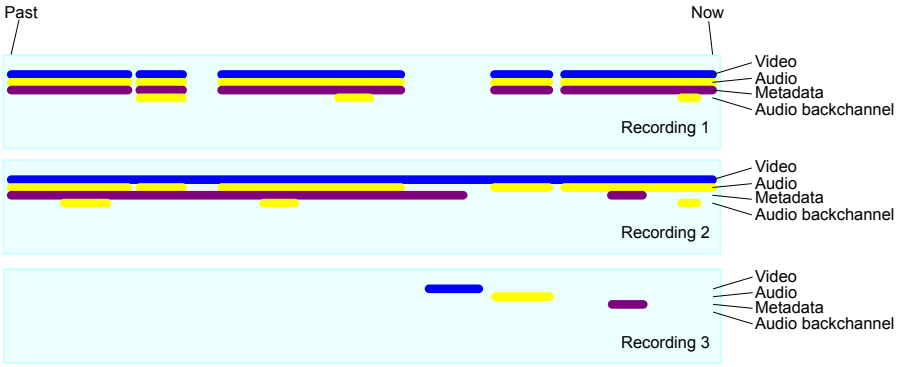

Figure 2: Example of recordings and tracks

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

At 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 may 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 like alarm recording or manual recording. A recording job with schedule token shall only be recording when the associated schedule is active.

For the cases where media attributes of a source are changed for an active recording job, the recording state is outside the scope of this specification.

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.

The ONVIF 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 ONVIF recording, the intent of the model in this standard 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 can also hold all the events generated by the data source (see section event handling of the ONVIF Core Specification and section on Metadata configuration in the ONVIF Media Service Specification). In addition, a device also conceptually record ONVIF defined historical events (see Recording Event Descriptions in the search service), this includes information like 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. This parameter has no effect on the data stored in external targets for e.g. Object Storage.

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* (see ONVIF Recording Search Service Specification) 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.

A device signaling support for recording to external targets via the SupportedTargetFormats capability shall support the target configuration with the following parameters:

Storage Token of a storage configuration.

Format Format of the recording. Examples are `MP4` and `CMAF`.

Prefix Path prefix to be inserted in the object key.

Postfix Path postfix to be inserted in the object key.

SpanDuration Maximum duration of a span.

SegmentDuration Maximum duration of a segment.

Encryption Optional encryption configuration.

StorageStrategy Strategy whether to directly record to external storage or to local storage or both in parallel.

See Annex B for ONVIF defined recording formats.

A device signaling support for SegmentExport shall support StorageStrategy modes Local and Both.

5.3.2 TrackConfiguration

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

The TrackConfiguration contains the following fields:

The **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* (see ONVIF Recording Search Service Specification) methods.

5.3.3 RecordingJobConfiguration

The RecordingJobConfiguration structure shall hold the configuration for a recording job. Its UML diagram is shown in Figure 3.

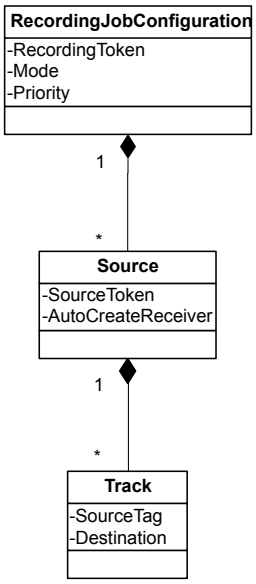

Figure 3: UML diagram of the RecordingJobConfiguration

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 shall only store 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 multiple recording jobs with the same highest priority it is undefined which of them is activated.

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

ScheduleToken: This attribute adds an additional requirement for activating the recording job. If this optional field is provided the job shall only record if the schedule exists and is active.

EventFilter: This set of parameters allows to control recording depending on a given set of event conditions.

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. If Type is http://www.onvif.org/ver10/schema/Receiver, the token is a ReceiverReference. 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, provisioned via the ONVIF Media Service or the ONVIF Media2 Service, instructing the device to obtain data from a profile that exists on the local device.

A device that includes the ONVIF Media Service or ONVIF Media2 service shall support a Media Profile token. A device that includes the ONVIF Receiver Service shall support a Receiver token.

AutoCreateReceiver: If a request includes this field set to true and no source token is provided, the device shall create a receiver object (through the receiver service) and assign the ReceiverReference to the **SourceToken** field. A device shall never report this parameter in a RecordingJobConfiguration. A device may reject a request that neither contains a SourceToken nor AutoCreateRecevier set to true.

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 case multiple RecordingJobs with differing Source are recording to the same Track it is undefined which of them is reported in the corresponding TrackInformation of the RecordingSearch API.

5.3.4 Event recording

A device signalling support for EventRecording via its capabilities shall support controlling recording job activity via the EventFilter with the following set of parameters:

Filter One or more filter pairs containing a mandatory topic filter as defined in section 9.6.3 of the ONVIF Core Specification. It may be associated with an optional message content filter as defined in section 9.4.4 of the ONVIF Core Specification.

Before Optional timespan to record before the actual event condition became active.

After Optional timespan to record after the actual event condition becomes inactive.

A device shall support filtering on topics and message source parameters. Filtering on message data values doesn't need to be supported since it may cause malfunctions.

A device shall support Before and After durations when their limit is signalled via the respective capability. A device may adapt the Before and After duration values to internal quantization.

A device shall at least record the event duration and the specified before and after timesspans. Due to the nature of GOP structures it may record more.

Note that non-property events result in an infinite short timespan. In such cases at least one I-Frame shall be recorded, optionally extended by before and after timespans.

A recording job of a device supporting both EventFilter and ScheduledRecording shall become active if both conditions are met.

5.4 CreateRecording

CreateRecording shall create a new recording.

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.
- **env:Sender - ter:InvalidArgVal - ter:AmbiguousConfiguration**
  Key/KID and AsymmetricEncryption are mutually exclusive configuration parameters
- **env:Receiver - ter:Action - ter:NoLocalRecording**
  Local recording not possible due to e.g. missing medium.
- **env:Receiver - ter:Action - ter:UnsupportedStorageStrategy**
  Storage strategy not supported.

ACCESS CLASS:

ACTUATE

When successfully completed, the device shall have created one or more tracks with the following configurations:

Table 2: Track configurations

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 to the 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.

This method has no effect on the data stored in external targets for e.g. Object Storage S3.

REQUEST:

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

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoRecording**
  The 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

`Key/KID` and `AsymmetricEncryption` elements of the encryption entry shall be mutually exclusive.

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.
- **env:Sender - ter:InvalidArgVal - ter:AmbiguousConfiguration**
  Key/KID and AsymmetricEncryption are mutually exclusive configuration parameters
- **env:Receiver - ter:Action - ter:NoLocalRecording**
  Local recording not possible due to e.g. missing medium.
- **env:Receiver - ter:Action - ter:UnsupportedStorageStrategy**
  Storage strategy not supported.

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**
  The 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 the TrackToken is unique within the recoding to which the new track belongs.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoRecording**
  The 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**
  The 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.

This method has no effect on the data stored in external targets for e.g. Object Storage S3.

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**
  The RecordingToken does not reference an existing recording.
- **env:Sender - ter:InvalidArgVal - ter:NoTrack**
  The 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**
  The RecordingToken does not reference an existing recording.
- **env:Sender - ter:InvalidArgVal - ter:NoTrack**
  The 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 configuration for the track.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoRecording**
  The RecordingToken does not reference an existing recording.
- **env:Sender - ter:InvalidArgVal - ter:NoTrack**
  The 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.

A device should reject a configuration that neither includes a source with a source token nor AutoCreateReceiver set to true.

If the configuration does not include any tracks a device should assign all tracks of the corresponding recording.

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 existing recording.
- **env:Sender - ter:InvalidArgVal - ter:BadConfiguration**
  The contents of the 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

If a request with AutoCreateReceiver is accepted the response shall provide the attached receiver token and the AutoCreateReceiver field shall be omitted.

The device response shall include the complete JobConfiguration including the associated job token and the resulting recording track configuration.

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**
  The 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 the 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**
  The JobToken does not reference an existing job.
- **env:Sender - ter:InvalidArgVal - ter:BadConfiguration**
  The contents of the 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**
  The 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**
  The 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**
  The JobToken does not reference an existing job.

ACCESS CLASS:

READ_MEDIA

The UML representation of the RecordingJobStateInformation structure is:

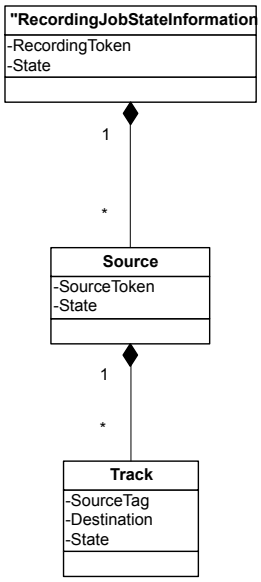

Figure 4: RecordingToken shall be the identification of the recording that the recording job records to.

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 aggregate state

Table 3: state rules

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 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/Media2 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 spare attributes the device signals that no track can be added.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoRecording**
  The 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. A client can also use AsyncronousOperationStatus event to monitor the progress of 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 progress of exported files via optional FileProgressStatus (tt:ArrayOfFileProgress) element, containing an array of file name and completion percentage of file upload from 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 file upload. The optional FileProgressStatus element is sent in 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 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

ACCESS CLASS:

READ_MEDIA

The requested operation does not exist.

5.23 GetExportRecordedDataState

GetExportRecordedDataState returns the status of export operations. This interface allows 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 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

ACCESS CLASS:

READ_MEDIA

The requested operation does not exist.

5.24 OverrideSegmentDuration

OverrideSegmentDuration dynamically updates a RecordingConfiguration to produce smaller recording segments for a limited time. This allows for a client to request faster access to recorded data while reducing the overall storage costs during normal operation.

When the temporary override is active, the device should attempt to close its currently recording segment and then shall record further segments with the new duration provided in this command. The override shall be active for the duration specified in the command, once the command expires the device shall resume recording with the original segment duration on its next segment. The override duration shall not exceed 1 hour.

When a new override request is sent before the previous override has expired, the new override shall replace the previous override and the new expiration shall apply.

The active override value, if any exists, shall be shown in the corresponding `RecordingConfiguration` and may not persist after a device restart. If the recording configuration is modified while an override is active, the override shall be removed.

A device that indicates support for OverrideSegmentDuration shall support this command.

REQUEST:

- **TargetSegmentDuration [xs:duration]**
  The new segment duration to be used for the duration of the override request.
- **Expiration [xs:duration]**
  The duration for which the override request shall be active.
- **RecordingConfiguration [tt:RecordingReference]**
  The recording configuration that needs to use the new segment duration.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoRecording**
  The RecordingConfiguration does not reference an existing recording.
- **env:Sender - ter:InvalidArgVal - ter:BadDuration**
  The value of the duration parameter is invalid.
- **env:Receiver - ter:ActionNotSupported - ter:NotImplemented**
  This optional method is not implemented.

ACCESS CLASS:

ACTUATE

5.25 GetServiceCapabilities

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

REQUEST:

This is an empty message.

RESPONSE:

- **Capabilities [trc:Capabilities]**
  List of capabilities as defined below.

FAULTS:

None

ACCESS CLASS:

PRE_AUTH

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 http://www.iana.org/assignments/media-types/media-types.xhtml. 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 ExportRecordedData command for the listed export file formats. A device shall only return this capability if it contains at least one export file format value. The value of ‘ONVIF’ refers to ONVIF Export File Format Specification.
ScheduledRecording	Indication that the device supports scheduled recording.
EventRecording	Indication that the device supports event triggered recording.
BeforeEventLimit	Indicates that the device supports before event durations up to the given value.
AfterEventLimit	Indicates that the device supports after event durations up to the given value.
SupportedTargetFormats	List of formats supported by the device for recording to an external target. See tt:TargetFormat for a list of definitions.
EncryptionEntryLimit	Number of encryption entries supported per recording. By specifying multiple encryption entries per recording, different tracks can be encrypted with different configurations.
SupportedEncryptionModes	Supported encryption modes. See tt:EncryptionMode for a list of definitions.

OverrideSegmentDuration Indicates that the device supports the OverrideSegmentDuration command.

AsymmetricEncryptionSupported Indicates that the device supports asymmetric encryption.

OnboardStorage Indicates if the device supports recording to onboard storage, default value is true.

If false, device shall support recording to an external target.

SegmentExport Indicates support for exporting recorded segments.

5.26 ListRecordedSegments

Lists available recorded segments related to the specified RecordingToken. A device signaling support for SegmentExport shall support this function.

The device shall provide results in StartTime ascending order. The device can decide to send less results if it needs to. The device shall indicate if there are more segments available in the time range by having the HasMoreResults field set to True.

When the HasMoreResults field is set to True in the response, the client shall adapt its next query by changing the From field to the last EndTime received from the previous request. The client should continue until the HasMoreResults field is set to False.

The listed segments shall have a stable start and end time as defined in C.4.

REQUEST:

- **Time [tt:DateTimeRange]**
- **RecordingToken [tt:RecordingReference]**
- **MaxResults - optional [xs:int]**
  The maximum number of results to return in one response.

RESPONSE:

- **Segment optional, unbounded [trc:Segment]**
  Contains a collection of segments with some basic metadata associated with the recording.
- **HasMoreResults [xs:boolean]**
  A flag indicating that more segments are available in the specified time range.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoRecording**
  The RecordingToken does not reference an existing recording.
- **env:Sender - ter:InvalidArgVal - ter:NoStorage**
  The TargetConfiguration does not reference an existing storage configuration.

ACCESS CLASS:

READ_MEDIA

5.27 ExportRecordedSegments

Request an export for the specified time range from existing locally recorded data. As such an operation can take long time, this method immediately returns an operation token. The device shall emit events as defined in section 8.9.7 Asynchronous Operation Status of the ONVIF Core specification. The device shall include the Alias parameter in the event if provided in the request. The Initialized event shall be emitted when returning the operation token. Progress updates shall be sent to inform clients about the progress of the export. Finally a Deleted event shall be emitted to signal completion or aborting of the operation

when StopExportRecordedSegments has been executed. A device signaling support for SegmentExport shall support this function.

An event shall be raised after each segment is uploaded to the Object Storage, which means the ArrayOfFileProgress shall have only a single element each time. The FileProgress element shall have the FileName element be the path as defined in B.4 appended to the StorageUri from the StorageConfiguration.

By default the segments are exported to the storage that is configured for the recording. When the parameter StorageToken is provided this target will be used.

Note that a client may subscribe to tns1:Monitoring/AsynchronousOperationStatus for enumeration and monitoring of active and queued exports.

REQUEST:

- **Time [tt:DateTimeRange]**
- **RecordingToken [tt:RecordingReference]**
- **Alias - optional [xs:string]**
  If provided, shall be included in the Monitoring/AsynchronousOperationStatus event.
- **StorageToken - optional [tt:ReferenceToken]**
  If provided, overrides the Target's storage configuration.
- **Track - optional [tt:TrackType]**
  An optional field if specified indicates which track is to be exported.

RESPONSE:

- **OperationToken [tt:ReferenceToken]**

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoRecording**
  The referenced recording does not exist.
- **env:Sender - ter:InvalidArgVal - ter:InvalidStorage**
  The referenced storage configuration does not exist or cannot be used.
- **env:Sender - ter:InvalidArgVal - ter:BadDuration**
  The value of the time parameter is invalid.

ACCESS CLASS:

READ_MEDIA

5.28 StopExportRecordedSegments

Stops the selected ExportRecordedSegments operation. A device signaling support for SegmentExport shall support this function.

REQUEST:

- **OperationToken [tt:ReferenceToken]**

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoOperation**
  The referenced operation does not exist.

ACCESS CLASS:

READ_MEDIA

5.29 Events

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 ONVIF Recording Search Service Specification.

5.29.1 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>
```

The ElementItem Information shall be provided whenever the state of the different tracks is not unique. It can be omitted when the state of all tracks of a recording is consistent.

5.29.2 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>
```

5.29.3 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>
```

This event is not applicable for data removal from object storage targets for e.g. Object Storage S3.

5.29.4 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>
```

5.30 Examples

5.30.1 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.

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

This example is very similar to example 1. The jobconfiguration 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)
```

Annex A.
Example scenario for Recording Job Priority (Informative)

This annex describes a scenario for Multiple Recording Jobs configured to record data into a single recording.

Accordingly, a device supporting Multiple Recording Jobs is required to change the Job Modes of Recording Jobs with respect to Priority, as described below :

Step 1 A Recording Job ‘J1’ with Priority ‘2’ is created in 'Active' mode

Job State of Recording Jobs after Step 1:

Recording Job ‘J1’ = ACTIVE

Step 2 A new Recording Job ‘J2’ with Priority ‘4’ is now created in 'IDLE' mode

Job States of Recording Jobs after Step 2:

Recording Job ‘J1’ = ACTIVE

Recording Job ‘J2’ = IDLE

Step 3 Another Recording Job ‘J3’ with higher Priority ‘3’ is now created in 'Active' mode. Because it has a higher priority than J1, it takes precedence.

Job States of Recording Jobs after Step 3:

Recording Job ‘J1’ = IDLE

Recording Job ‘J2’ = IDLE

Recording Job ‘J3’ = ACTIVE

Step 4 Job States of Recording Jobs after Step 4:

Recording Job ‘J3’ is now deleted. 'J1' becomes active again since it is the active job with the highest priority.

Recording Job ‘J1’ = ACTIVE

Recording Job ‘J2’ = IDLE

Annex B.
Object storage recording format (Normative)

B.1 Overview

This annex describes the requirements for recording from a device to an object storage for later use by a receiver. The device could be a camera, the receiver could be a video player. Figure B.1 shows the data flow for object storage recording.

Figure B.1: Data flow for object storage recording

The device uploads recorded media as objects containing fragmented MP4 data according to ISO/IEC 14496-12, the ISO base media file format. In this annex, a media object containing fragmented MP4 data is called a segment. Each segment starts with a header containing initialization data, usually consisting of a `FileTypeBox` and a `MovieBox`. Each segment contains one or more fragments with recorded media, usually consisting of a `MovieFragmentBox` and a `MediaDataBox`. Each segment contains recorded media of a continuous period of time.

Multiple successive segments without a time gap in between form a span. In this way, the size of segments can be limited when recording continuously for a longer period of time.

For a single span, only a single segment contains recorded media of one point in time. Depending on the recording format, video, audio, and metadata are written into separate segments. In this case, the segments of each media type form separate spans. Figure B.2 shows the logical model for object storage recording.

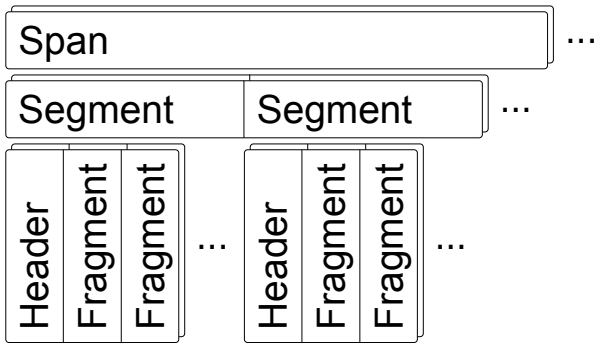

Figure B.2: Logical model for object storage recording

B.2 Spans

When a recording is activated, the device shall open new spans for this recording according to the rules of the configured `Format`. Inversely, when a recording is deactivated, the device shall close all open spans of this recording. When a `SpanDuration` is configured for a recording, the recording is active, and the duration of a span of this recording reaches the configured `SpanDuration`, the device shall close all open spans of the recording and open new spans according to the rules of the configured `Format`.

All fragments of a span shall be independently processable and presentable with the information of a `MovieBox` in any of the segments of the span. If the information in the `MovieBox` required to process the fragments needs to change while the recording is active, the device shall close all open spans of the recording and start new spans according to the rules of the configured `Format`.

B.3 Segments

When a span is opened, the device shall also open a new segment for this span. Inversely, when a span is closed, the device shall close the open segment of this span. When a span is open and the duration of the open segment of this span reaches the `SegmentDuration` configured for the recording, the device shall close the open segment of this span and open a new segment for this span. When closing the open segment of a span and opening a new segment for this span, there shall be no time gap between the two segments. Note that the actual segment duration may vary and can be plus or minus one GOP size from the configured duration because a segment shall start with an I-Frame. Depending on resource constraints, the device may close the segment earlier than the configured `SegmentDuration` if the requested length cannot be achieved. This might happen, for example, if the device runs out of available memory to generate the segment.

The device shall generate each segment as a fragmented MP4 file according to ISO/IEC 14496-12, the ISO base media file format. A segment shall contain recorded media for the time it was open. The recording formats provide specification about segment types, their contents, and restrictions on the ISO base media file format in the following sections.

B.4 Segment objects

The device shall upload each segment as one object, identified by its object key, to the object storage. The device shall generate the object key for a segment according to the following ABNF `key` rule (according to RFC 5234):

```
key      = lead "/" time "." counter "." extension
lead    = [prefix "/"] date ["/" postfix]
counter = 1*DIGIT
date    = year "-" month "-" day
time    = hour "/" minute "-" second [secfrac] "Z"
year    = 4DIGIT
month   = 2DIGIT ; 01-12
day     = 2DIGIT ; 01-31
hour    = 2DIGIT ; 00-23
minute  = 2DIGIT ; 00-59
second  = 2DIGIT ; 00-60 (due to leap second rules)
secfrac = "." 1*6DIGIT
```

If a `Prefix` is configured for the recording, the device shall include `prefix` in `lead` and replace it with the configured `Prefix`. If a `Postfix` is configured for the recording, the device shall include `postfix` in `lead` and replace it with the configured `Postfix`. The device shall replace `date` and `time` respectively with the date and time portion of the first sample contained in the segment in Universal Coordinated Time (UTC), following the ABNF rules:

- `year` represents the 4-digit year
- `month` represents the 2-digit month (01 to 12)
- `day` represents the 2-digit day of the month (01 to 31)
- `hour` represents the 2-digit hour (00 to 23)
- `minute` represents the 2-digit minute (00 to 59)
- `second` represents the 2-digit second (00 to 60, due to leap second rules)
- `secfrac` represents the optional fractional seconds with a precision of up to 6 digits

The device shall replace `counter` with a span-dependant counter set to 0 for a new span and incremented by 1 when closing a segment of this span. The device shall replace `extension` with the extension defined by the type of span the segment belongs to.

Note: `date` is conformant to the extended date format defined in ISO 8601-1. `time` is **not** conformant to the extended time format defined in ISO 8601-1.

B.5 Span end objects

To store the end time of a span, the device should store an additional empty object when closing a span. The device shall generate the object key for this object according to the ABNF `key` rule and replacement rules defined in B.4. The device shall replace `date` and `time` respectively with the date and time portion of the end of the last sample contained in the span in Universal Coordinated Time (UTC), following the ABNF rules. The device shall replace `counter` by a value equal to the counter of the last segment of the span plus 1. The device shall replace `extension` according to the following ABNF `extension` rule (according to RFC 5234):

```
extension = span-extension "_end"
```

The device shall replace `span-extension` with the extension defined by the span type.

B.6 Recording format MP4

If the `Format` configured for a recording is multiplexed `MP4`, the device shall follow the rules given in this section.

The device shall open a single span for an active recording containing all tracks of the recording. The device shall use `mp4` as the extension for this span type. The device shall upload each segment object belonging to a span of this type with the content type `video/mp4`. This content type is also used for audio-only recordings or metadata-only recordings. The device should include the `codecs` parameter in the content type according to RFC 6381.

The `MovieBox` contained in a segment shall not reference any samples. A segment shall contain a `MovieFragmentRandomAccessBox` as the last box. The `MovieFragmentRandomAccessBox` shall provide references for all tracks and their applicable `MovieFragmentBox` instances.

B.7 Recording format CMAF

If the `Format` configured for a recording is `CMAF`, the device shall follow the rules given in this section.

The device shall open up to three spans for an active recording, each containing a single track of the recording.

If the recording contains a video track, the device shall open a video span containing this track. The device shall use `m4v` as the extension for this span type. The device shall upload each segment object belonging to a span of this type with the content type `video/mp4`.

If the recording contains an audio track, the device shall open an audio span containing this track. The device shall use `m4a` as the extension for this span type. The device shall upload each segment object belonging to a span of this type with the content type `audio/mp4`.

If the recording contains a metadata track, the device shall open a metadata span containing this track. The device shall use `m4m` as the extension for this span type. The device shall upload each segment object belonging to a span of this type with the content type `application/mp4`.

The device should include the `codecs` parameter in the content type according to RFC 6381 for all span types defined in this section.

The device shall generate each segment according to ISO/IEC 23000-19, the common media application format (CMAF) for segmented media, with one CMAF header and one or more CMAF fragments. A segment may contain a `MovieFragmentRandomAccessBox` as the last box. If a segment contains a `MovieFragmentRandomAccessBox`, every CMAF fragment shall be referenced in the `MovieFragmentRandomAccessBox`.

Note: As the different tracks are contained in different spans, the segments do not necessarily align. This is illustrated in Figure B.3.

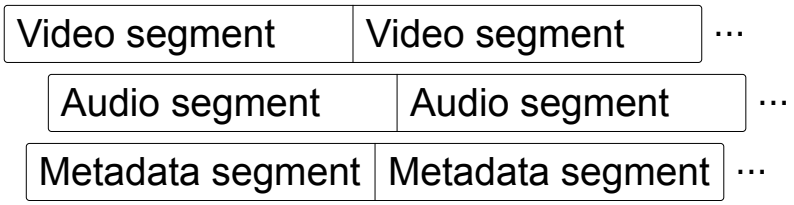

Figure B.3: Alignment of different segment types

B.8 Encryption

A device signaling support for encrypted recording via `SupportedEncryptionModes` shall support creating encrypted segments according to [ISO/IEC 23001-7]. Each `Encryption` entry configured for a recording covers a distinct set of tracks for which to apply the encryption, identified by the `Track` element. If an encryption entry does not contain any `Track` element, all tracks of the recording shall be encrypted using the same encryption entry. If an encryption entry contains one or more `Track` elements, specified tracks indicated by the track tokens of the recording shall be encrypted using the encryption entry. The device shall encrypt the tracks with the scheme given by the `Mode` configured for the encryption entry:

CENC AES-CTR mode full sample and video NAL Subsample encryption, defined in ISO/IEC 23001-7.

CBCS AES-CBC mode partial video NAL pattern encryption, defined in ISO/IEC 23001-7.

The device shall create a unique initialization vector for each fragment present in a segment. Each encrypted segment shall include the `moov` box containing the required PSSH box(es) according to the encryption entry configuration. If an encryption entry is reconfigured for an active recording, the device shall continue to use the old configuration for any open segments and shall start to use the new configuration for new segments.

B.8.1 W3C Common Encryption System

All encrypted segments, in both supported modes and with either KID/Key strategy or AsymmetricEncryption strategy, shall include one PSSH box following the [W3C 'cenc' Initialization Data Format] Common SystemID, containing the KIDs of the keys used for encryption. Additional types of 'PSSH' boxes may be present to support alternative DRM systems, such as the Asymmetric Key System.

B.8.1.1 Syntax

```
aligned(8) class ProtectionSystemSpecificHeaderBox extends FullBox('pssh', version=1,
                                                                    flags=0)
{
  unsigned int(8)[16] SystemID;
  unsigned int(32) KID_count;
  for (i=1; i <= KID_count; i++) {
    unsigned int(8)[16] KID;
  }
  unsigned int(32) DataSize = 0;
}
```

See W3C "cenc" Initialization Data Format [https://www.w3.org/TR/eme-initdata-cenc] for an example of this box.

B.8.1.2 Semantics

`Version` shall be '1'.

`Flags` shall be '0'.

`SystemID` shall be `'1077efec-c0b2-4d02-ace3-3c1e52e2fb4b'`.

`KID_count` be equal to the number of different keys used in the segment.

`KID`, one for each `RecordingEncryption`.

`DataSize` shall be '0'.

B.8.2 Asymmetric key system

When `AsymmetricEncryption` is set, the device shall generate its own key and encrypt each segment with it. A KID shall also be generated for use in the PSSH boxes. If the `KeyRotationDuration` is set, then the device shall generate a new Key and KID at the specified time interval, but still use the current key until the segment is finished. New segments shall use the latest generated `Key` for its encryption.

In addition to the W3C Initialization Data, the device shall also include in each segment an `AsymmetricKeySystemHeaderBox` PSSH box containing the information needed to play the segment. This box contains the KID, the encrypted symmetric key and the list of certificates that can be used to decrypt it. The symmetric key is encrypted once for each configured certificate using their public key. The client needs access to at least one of the certificates' private key to decrypt it, either directly or through a key server. The client can then decrypt the frames using this symmetric key. If tracks are encrypted using different keys, then one `AsymmetricKeySystemHeaderBox` shall be present per `KID`.

Additional 'PSSH' boxes may be present to support alternative DRM systems. Note that DRM systems and client implementations are outside the scope of this specification.

B.8.2.1 Syntax

```
aligned(8) class AsymmetricKeySystemHeaderBox extends FullBox('pssh', version=1, flags=0)
{
  unsigned int(8)[16] SystemID;
  unsigned int(32) KID_count = 1;
  unsigned int(8)[16] KID;
  unsigned int(32) DataSize;
  unsigned int(32) EncryptedKeyEntryCount;
  for (i=1; i <= EncryptedKeyEntryCount; i++) {
    unsigned int(16) CertificateThumbprintAlgorithm;
    unsigned int(32) CertificateThumbprintSize;
    unsigned int(8)[CertificateThumbprintSize] CertificateThumbprint;
    unsigned int(8) EncryptionVersion;
    if (EncryptionVersion == 1) {
      unsigned int(16) EncryptedSymmetricKeySize;
      unsigned int(8)[EncryptedSymmetricKeySize] EncryptedSymmetricKey;
    } else if (EncryptionVersion == 2) {
      unsigned int(16) HpkeKem;
      unsigned int(16) HpkeKdf;
      unsigned int(16) HpkeAead;
      unsigned int(8)[EncapsulatedSharedSecretSize] EncapsulatedSharedSecret;
      unsigned int(16) EncryptedSymmetricKeySize;
      unsigned int(8)[EncryptedSymmetricKeySize] EncryptedSymmetricKey;
      unsigned int(16) InfoSize;
      unsigned int(8)[InfoSize] Info;
    }
  }
}
```

B.8.2.2 Semantics

`Version` shall be '1'.

`Flags` shall be '0'.

`SystemID` shall be `'a4852bd0-80fc-484e-b9e1-78a74d49f5ce'`.

`KID_count` shall be '1'.

`KID` is the generated UUID by the device. This `KID` should be the same in all segments until the symmetric key changes.

`DataSize` is the size in bytes of all the other fields present in this box following this field.

`EncryptedKeyEntryCount` Number of entries containing encryption information required to decrypt the symmetric key. Shall be 1 or more.

`CertificateThumbprintAlgorithm` defines the algorithm identifier used to compute the certificate thumbprint. The valid identifiers are `1` for SHA-1, `2` for SHA-256.

`CertificateThumbprintSize` Size of the `CertificateThumbprint` field.

`CertificateThumbprint` Thumbprint of the certificate used to encrypted the symmetric key.

`EncryptionVersion` defines the encryption strategy used to encrypt the symmetric key for this certificate.

`HpkeKem` defines the KEM algorithm identifier according to IANA used to encrypt the key with the current certificate.

`HpkeHkdf` defines the HKDF algorithm identifier according to IANA used to encrypt the key with the current certificate.

`HpkeAead` defines the AEAD algorithm identifier according to IANA used to encrypt the key with the current certificate.

`EncapsulatedSharedSecretSize` is implicitly defined to the `Nenc` parameter of the `HpkeKem` algorithm.

`EncapsulatedSharedSecret` is the HPKE shared secret value necessary to decrypt the encrypted key according to RFC 9180.

`EncryptedSymmetricKeySize` Size of the `EncryptedSymmetricKey` field. Valid values depend on the encryption algorithm used by the certificate.

`EncryptedSymmetricKey` The symmetric key (identified by `KID`) used for frame encryption, encrypted using the public key of the certificate according to the encryption version.

`InfoSize` is the size in bytes of the `Info` field.

`Info` is the value configured by the `AdditionalInfo` configuration field encoded as UTF-8.

B.8.2.3 Encryption Version 1

This version is defined for use when the certificate contains an RSA public key. When using this version, the symmetric key is directly encrypted using the public key of the certificate and the RSA-OEAP padding scheme as defined in [RFC 5652].

B.8.2.4 Encryption Version 2

This version is defined for the use of the HPKE algorithm as defined in [RFC 9180]. It is used when the certificate contains an EC public key. Using the public key of the certificate and the algorithms defined in the `HpkeKem`, `HpkeHkdf`, and `HpkeAead` fields, the `EncapsulatedSharedSecret` field is derived using the `Base` mode of HPKE.

The `AdditionalInfo` field of the `AsymmetricEncryption` configuration is to be used as the `info` parameter of the HPKE key derivation function. If the field is missing, the device shall use an empty string.

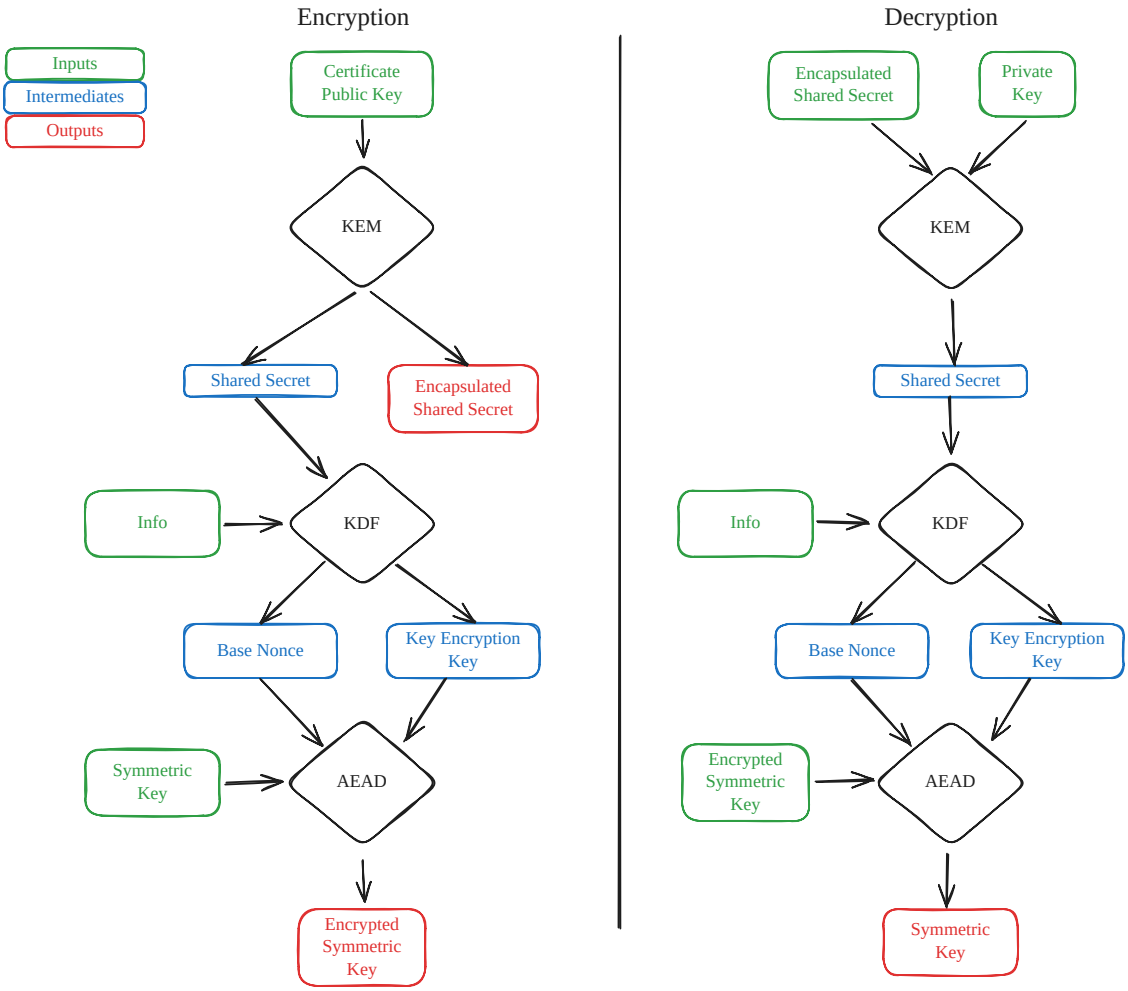

Figure B.4: Encryption (Left) and Decryption (Right) using the HPKE algorithm

B.9 Object attributes

The device may store additional attributes defined in Table B.1 as object metadata when uploading segment objects.

Note: When sending attributes as metadata HTTP headers, the attribute keys need to be prefixed with a value determined by the type of storage configured for the recording. If the storage is of type `ObjectStorageS3`, the attribute keys need to be prefixed with `x-amz-meta-` (for example `x-amz-meta-Version: 1.0`). If the storage is of type `ObjectStorageAzure`, the attribute keys need to be prefixed with `x-ms-meta-` (for example `x-ms-meta-Version: 1.0`). Audio-related attributes may be stored for segment objects containing only a video track and video-related attributes may be stored for segment objects containing only an audio track.

Table B.1: Object attributes

Key	Value	Format
`Version`	`1.0`	`major "." minor`
`Source`	Identifier of the physical device	String
`EventTopic`	Topic of the event that triggered the recording job	String
`Duration`	Exact duration of the segment. Sum of the duration of each frame in the file, if	ISO 8601-1

Key	Value	Format
	there are multiple tracks the video track shall be used if present.	
FullContentType	Combined content type of all media types contained in the recording	video/mp4 content type with codecs parameter according to RFC 6381
VideoChannel	Video channel identifier	String
VideoBitrate	Video bitrate in kilobits per second	Integer
VideoWidth	Video width in pixels from VisualSampleEntry	Integer
VideoHeight	Video height in pixels from VisualSampleEntry	Integer
VideoFramerate	Average video framerate in frames per second	Float
AudioChannel	Audio channel identifier	String
AudioBitrate	Audio bitrate in kilobits per second	Integer
AudioSampleRate	Audio sample rate in kilohertz	Integer
Timescale	The timescale (or clock) from the segment in hertz	Integer (See B.9.1)
SeekTable	Seek table to efficiently locate file fragments and seek in the file	String (See B.9.2)

For storage target systems that do not allow attribute changes during write the attribute Duration should be omitted when the total duration of the file may change while it is written e.g. due to an override segment duration request.

B.9.1 Timescale Attribute

The value of the timescale property of the MediaHeaderBox in the segment as defined in ISO/IEC 14496-12:2012 section 8.4.2. If there are multiple tracks in the segment each of these boxes shall all have the same timescale value. The MovieHeaderBox shall also have the same timescale value.

B.9.2 SeekTable Attribute

This seek table attribute is a string that contains the 'mfra' (Movie Fragment Random Access) box corresponding to the associated ISO file. It is encoded in base64 with padding according to RFC 4658. The binary format of the box is defined in ISO/IEC 14496-12:2012 section 8.8.9. This optional box has to be the last one at the end of the file. The optionally condensed copy allows the receiver to efficiently locate file fragments and seek in the file without the need to access the file itself.

The size of the SeekTable should be limited to a reasonable access precision in the order of single digit seconds. Additionally the SeekTable should contain no more than twenty entries. When the SeekTable attribute is present it shall contain at least one entry for the very first sample of each track. The mfro box may be omitted in the SeekTable as it doesn't add any value, but the mfra length shall be updated accordingly.

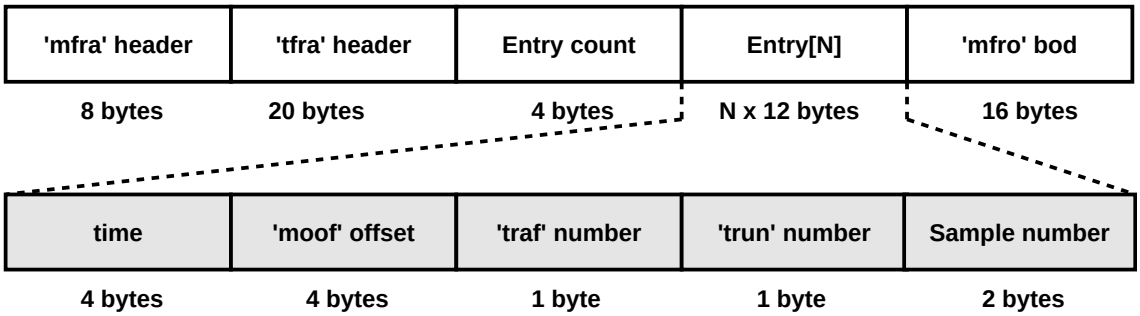

Figure B.5: Typical structure of a Movie Fragment Random Access ('mfra') box

The Timescale attribute shall be provided if the SeekTable attribute is present, as it is necessary to interpret the SeekTable.

B.10 Examples

Given the following storage configuration in a recording configuration:

```
<tt:Target>
  <tt:Storage>1</tt:Storage>
  <tt:Format>MP4</tt:Format>
  <tt:Prefix>site-2</tt:Prefix>
  <tt:Postfix>camera-5</tt:Postfix>
  <tt:SegmentDuration>PT60S</tt:SegmentDuration>
</tt:Target>
```

The key of a segment object starting on 2022-11-08 at 15:08:23.547 (UTC) is:

```
site-2/2022-11-08/camera-5/15/08-23.547Z.0.mp4
```

The key of a span end object for a span consisting of 85 segments and ending on 2022-11-08 at 16:34:11.421 (UTC) is:

```
site-2/2022-11-08/camera-5/16/34-11.421Z.85.mp4_end
```

Given the following storage configuration in a recording configuration:

```
<tt:Target>
  <tt:Storage>1</tt:Storage>
  <tt:Format>CMAF</tt:Format>
  <tt:Prefix>site-2</tt:Prefix>
  <tt:Postfix>camera-5</tt:Postfix>
  <tt:SegmentDuration>PT5S</tt:SegmentDuration>
</tt:Target>
```

The key of a video segment object starting on 2022-11-08 at 15:08:23.547 (UTC) is:

```
site-2/2022-11-08/camera-5/15/08-23.547Z.0.m4v
```

The key of a span end object for a video span consisting of 1020 segments and ending on 2022-11-08 at 16:34:11.421 (UTC) is:

```
site-2/2022-11-08/camera-5/16/34-11.421Z.1020.m4v_end
```

The key of an audio segment object starting on 2022-11-08 at 15:08:23.989 (UTC) is:

```
site-2/2022-11-08/camera-5/15/08-23.989Z.0.m4a
```

The key of a span end object for an audio span consisting of 1020 segments and ending on 2022-11-08 at 16:34:11.779 (UTC) is:

```
site-2/2022-11-08/camera-5/16/34-11.779Z.1020.m4a_end
```

The key of a metadata segment object starting on 2022-11-08 at 15:08:23.692 (UTC) is:

```
site-2/2022-11-08/camera-5/15/08-23.692Z.0.m4m
```

The key of a span end object for a metadata span consisting of 1020 segments and ending on 2022-11-08 at 16:34:11.871 (UTC) is:

```
site-2/2022-11-08/camera-5/16/34-11.871Z.1020.m4m_end
```

Annex C.
Segment export (Normative)

C.1 Overview

This annex describes the requirements for recording and exporting media streams in a cloud-friendly format. The device will need to record on its local storage when a `StorageStrategy` has been set to "Local" or "Both". While the device can record locally using any format, segments can be queried at any time and the resulting list of segments shall be stable. "Stable" here is defined in the sense that each frame will always be part of the same segment and the available segments are immutable.

Specific recorded segments can then be exported to the cloud on request and each exported segment shall yield the same result (a CMAF or MP4 file containing all its frames) on a successful export operation.

The deterministic properties of the segments (when queried or exported) facilitate features like HLS or MPEG-DASH playback from the device, and also a simple way to implement on-demand video trickling from the local storage for a device to the cloud.

C.2 Recording segments

There are three possible recording `StorageStrategy` defined for `RecordingTargetConfiguration`:

External The device will automatically upload each recorded segment to the Cloud location as soon as possible. After completion or on error the data won't be kept locally on the device. This is the default value if none has been set.

Local The device will record the segments on its local storage and respect the MaximumRetentionTime of the RecordingConfiguration.

Both The device will record the segments on its local storage and respect the MaximumRetentionTime of the RecordingConfiguration. It will also attempt to automatically upload each segment to the Cloud location as they become available.

When recording on its local storage, the device may need to delete the oldest recording segments if storage space is insufficient.

When uploading to the configured Cloud location, if an error occurs the device can retry but should abandon if the operation took too long and the next segment is ready to be uploaded. Only whole segments should be uploaded and committed to the cloud storage (no incomplete segment should be made available on failure).

C.3 Segment export commands

The `ListRecordedSegments` command lists the segments currently available for export from the device local storage. For a given time range, this command shall always return the same segments with two exceptions: the newly recorded segments are added as they become available, and the deleted segments are removed from it (for example if the device had to delete them to make room for the new ones). The results shall include segments that cover the requested time range entirely, even if they start before or end after the requested time range.

The `ExportRecordedSegments` command requests an export operation for a time-range of segments. A device shall accept multiple request but may queue and process them sequentially. Segments shall be uploaded in chronological order. Progress can be tracked via the AsynchronousOperationStatus event. Operations can be cancelled at any time with `StopExportRecordedSegments`. The segment names shall follow B.4.

Note: it is possible that the segments listed with `ListRecordedSegments` are not exactly the ones exported with `ExportRecordedSegments` as some segments could be deleted/added in-between the two commands being executed.

C.4 Segment stability

Segment stability is essential for this feature: each frame shall be part of exactly one segment and each segment shall always be immutable in time. Additionally, each segment shall start with a keyframe to make it independent from the previous one. Some devices may record frames in a proprietary format, so the segments won't exist until exported to the cloud storage. Since the device shall be able to list them beforehand, we suggest a simple algorithm to determine how many segments will be in each time range and which frames will be part of them.

The figure Figure C.1 illustrates how a global real time clock timestamp (that doesn't reset on device reboot or resync on NTP) could be segmented in time ranges for the target segment duration with a modulus operation. Then each boundary of these ranges are adjusted to match the time of the next keyframe. The result is the list of segment boundaries that overlap with the original time range of the query. Care should be taken not to return the most recent segment if it is incomplete (when more frames may be appended to it later, i.e.: it overlaps with the current time).

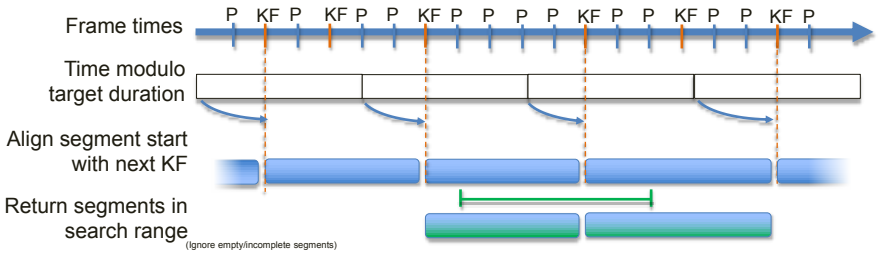

Figure C.1: Example of possible algorithm to ensure segment stability

Annex D.
Revision History

Rev.	Date	Editor	Changes
2.1	Jul-2011	Hans Busch	Split from Core 2.0 without change of content.
2.1.1	Jan-2012	Hans Busch	Change Requests 293, 297, 535
2.2	Apr-2012	Hans Busch	Change Requests 608, 625, 636, 673
2.2.1	Dec-2012	Hans Busch, Michio Hirai	Change Requests 708, 709, 719, 759, 827, 845, 852, 866, 867, 870, 862, 872, 861
2.3	May-2013	Michio Hirai	Change Request 934
2.4	Aug-2013	Michio Hirai	Change Request 1073, 1086
2.4.1	Dec-2013	Michio Hirai	Change Request 1148, 1189
2.4.2	Jun-2014	Michio Hirai	Change Request 1292, 1298, 1304, 1412
2.5	Dec-2014	Hasan Timucin Ozdemir	Added 5.21 ExportRecordedData command and corresponding capability flag in 5.24 Capabilities Added 5.22 StopExportRecordedData Added 5.23 GetExportRecordedDataStatus
16.12	Dec-2016	Hans Busch	Change Request 1991
17.06	Jun-2017	Stefan Andersson, Hiroyuki Sano	Update method layouts Change Request 1843, 2060, 2062 Change Request 2061, 2063, 2065, 2109
17.12	Dec-2017	Hiroyuki Sano	Change Request 2178, 2179, 2185
18.06	Jun-2018	Hiroyuki Sano	Change Request 2230, 2258
19.06	Jun-2019	Hans Busch	Added Scheduled Recording
21.12	Dec-2021	Sergey Bogdanov	Change the "role" attribute for request access class.
22.12	Dec-2022	Hans Busch	Add support for event recording.
23.06	Jun-2023	Felix Schuetz	Add annex on object storage recording.
24.12	Dec-2024	Sriram Bhetanabottla	Clarify delete interfaces for object storage.
25.06	Jun-2025	Jean-Francois Levesque, Jose Melancon	Add the possibility to temporary override of segment duration for Cloud Recording.
25.12	Dec-2025	Maxime Bédard, Jose Melancon, Hans Busch, Sriram Bhetanabottla	Add seektable and timescale attributes in cloud object metadata. Add frame Encryption for cloud recording.
26.06	Jun-2026	Jean-Francois Levesque, Jose Melancon, Sujith Raman, Sriram Bhetanabottla	Add support for listing and export of segments to an Object Storage. Add OnboardStorage and AsymmetricEncryptionSupported capabilities.