ONVIF™
ONVIF Recording Control Service Specification

Version 22.12

December, 2022

CONTENTS

1 Scope **5**

2 Normative references **5**

3 Terms and Definitions **5**

3.1 Definitions 5

4 Overview **5**

4.1 Storage 5
4.1.1 Storage Model 6
4.1.2 Recording 7

5 Recording control **7**

5.1 Introduction 7
5.2 General Requirements 8
5.3 Data structures 8
5.3.1 RecordingConfiguration 8
5.3.2 TrackConfiguration 9
5.3.3 RecordingJobConfiguration 9
5.3.4 Event recording 10
5.4 CreateRecording 11
5.5 DeleteRecording 11
5.6 GetRecordings 12
5.7 SetRecordingConfiguration 12
5.8 GetRecordingConfiguration 13
5.9 CreateTrack 13
5.10 DeleteTrack 14
5.11 GetTrackConfiguration 14
5.12 SetTrackConfiguration 15
5.13 CreateRecordingJob 15
5.14 DeleteRecordingJob 16
5.15 GetRecordingJobs 16
5.16 SetRecordingJobConfiguration 17
5.17 GetRecordingJobConfiguration 17
5.18 SetRecordingJobMode 18
5.19 GetRecordingJobState 18
5.20 GetRecordingOptions 20
5.21 ExportRecordedData 20
5.22 StopExportRecordedData 21
5.23 GetExportRecordedDataState 22
5.24 GetServiceCapabilities 22
5.25 Events 23
5.25.1 Recording job state changes 23

5.25.2 Configuration changes 23
5.25.3 Data deletion 24
5.25.4 Recording and track creation and deletion 24
5.26 Examples 25
5.26.1 Example 1: setup recording of a single camera 25
5.26.2 Example 2: Record multiple streams from one camera to a single recording 26
Annex A Example scenario for Recording Job Priority (Informative) 27
Annex B Revision History 28

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

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>

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.

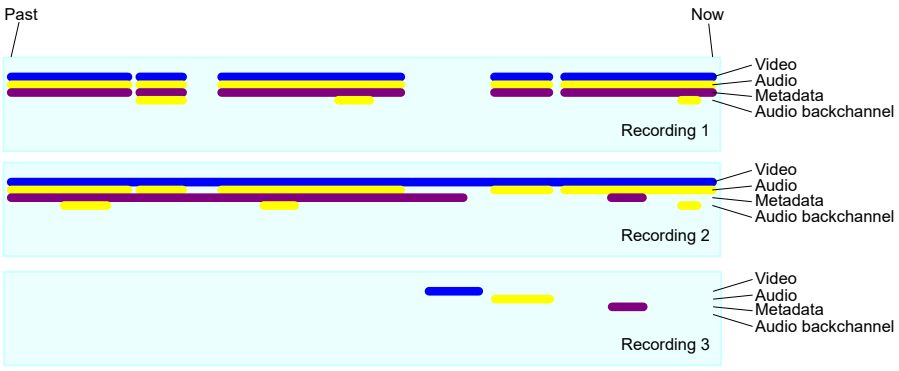

Figure 1: Storage Model with Tracks

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 infor-

mation 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

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.

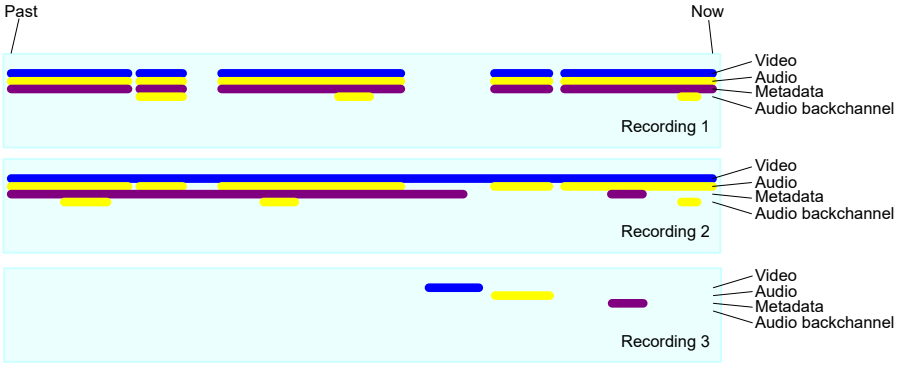

Figure 2: Example of recordings and tracks

The figure shows three recordings, each with a video, a metadata and two audio tracks. Here 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 ac-

cessible anymore. If the MaximumRetentionPeriod is set to 0, the device shall not limit the retention time of stored data, except by resource constraints. Whatever the value of MaximumRetentionTime, the device may automatically delete recordings to free up storage space for new recordings.

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

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 hown in Figure 3.

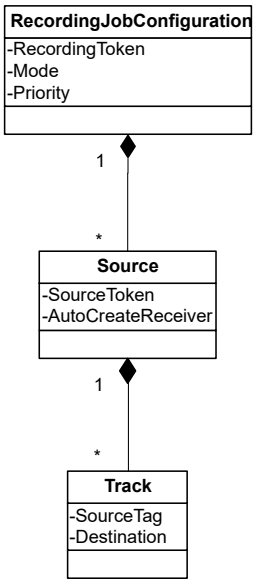

Figure 3: UML diagram of the RecordingJobConfiguration

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 will only store the data for the recording job with the highest priority. The priority is specified per recording job, but the device shall determine the priority of each track individually. If there are two recording jobs with the same priority, the device shall record the data corresponding to the recording job that was activated the latest.

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 condititions.

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, instructing the device to obtain data from a profile that exists on the local device.

A device that includes the ONVIF Media Service shall support a Media Profile token and a device that includes the ONVIF Receiver Service shall support a Receiver token.

If the **SourceToken** is omitted, **AutoCreateRecevier** shall be true.

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

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

Destination: The destination is the track token of the track to which the device shall store the received data. All tracks must 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 the RecordingSearch API.

5.3.4 Event recording

A device signalling support for EventRecording via its capabilities shall support controling 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.

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.

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

REQUEST:

- **RecordingToken [RecordingToken]**
  Identifies the recording that shall be changed.
- **RecordingConfiguration [RecordingConfiguration]**
  The new configuration for the recording.

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter: BadConfiguration**
  The configuration is invalid.

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

ACCESS CLASS:

ACTUATE

5.8 GetRecordingConfiguration

GetRecordingConfiguration shall retrieve the recording configuration for a recording

REQUEST:

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

RESPONSE:

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

FAULTS:

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

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 configurartion 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.

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

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

5.14 DeleteRecordingJob

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

REQUEST:

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

RESPONSE:

This is an empty message.

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:NoRecordingJob**
  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:

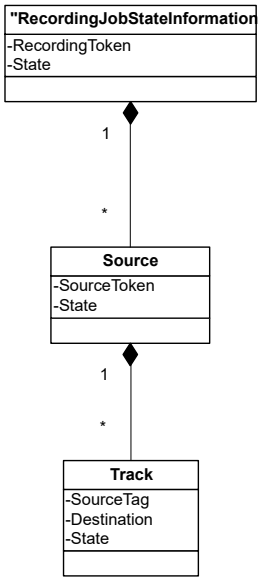

Figure 4: RecordingToken shall be the identification of the recording that the recording job records to.

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, containining 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 elemet 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 relavant 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 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 defintions 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 ony 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.

5.25 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.

5.25.1 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>
```

5.25.2 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>
```

5.25.3 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>
```

5.25.4 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>
```

5.26 Examples

5.26.1 Example 1: setup recording of a single camera

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

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

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

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

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

This completes the configuration step.

Finally, to really start recording, some entity calls

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

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

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

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

This example is very similar to example 1. 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.

As described in Section 5.3, "If there are two recording jobs with the same priority, the device shall record the data corresponding to the recording job that was activated the latest."

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 ‘1’ is created in 'Active' mode

Job Modes of Recording Jobs after Step 1:

Recording Job ‘J1’ = ACTIVE

Step 2 A new Recording Job ‘J2’ with Priority ‘1’ is now created in 'Active' mode

Job Modes of Recording Jobs after Step 2:

Recording Job ‘J1’ = IDLE

Recording Job ‘J2’ = ACTIVE

Step 3 Another Recording Job ‘J3’ with higher Priority ‘2’ is now created in 'Active' mode. Because it has a higher priority than J2, it takes precedence.

Job Modes of Recording Jobs after Step 3:

Recording Job ‘J1’ = IDLE

Recording Job ‘J2’ = IDLE

Recording Job ‘J3’ = ACTIVE

Step 4 Recording Job ‘J3’ is now deleted, 'J1' and 'J2' are both at the highest priority, so Section 5.14 applies, and the device can activate either 'J1' or 'J2'.

Job Modes of Recording Jobs after Step 4, possibility 1:

Recording Job ‘J1’ = ACTIVE

Recording Job ‘J2’ = IDLE

Job Modes of Recording Jobs after Step 4, possibility 2:

Recording Job ‘J1’ = IDLE

Recording Job ‘J2’ = ACTIVE

Annex B.
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 Capabilitites 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.