ONVIF™
Recording Control Service Specification

Version 2.2
May, 2012

CONTENTS

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-v220.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.

3.2 Abbreviations

ONVIF	Open Network Video Interface Forum

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.

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.

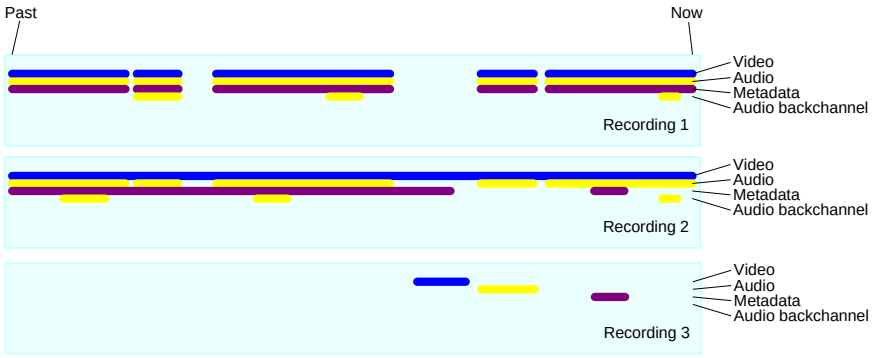


Figure 1: Storage Model with Tracks

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.

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.

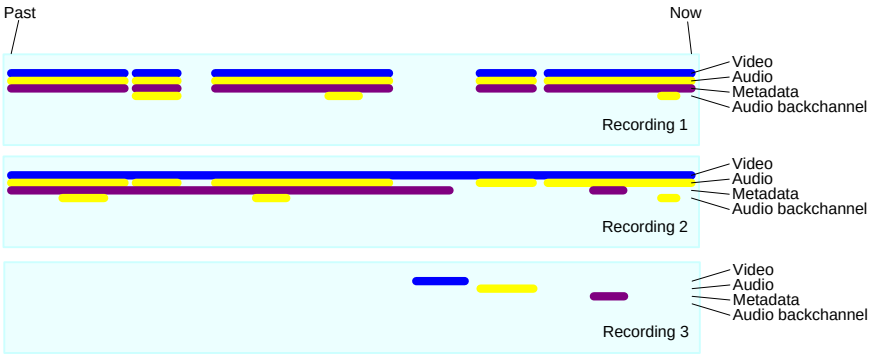


Figure 2: Example of recordings and tracks

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.

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

None of the other fields defined in this structure shall be used by the device. Instead, it simply stores this information, and it shall return it through the *GetRecordingConfiguration* and *GetRecordingInformation* (see ONVIF Recording Search Service Specification) methods.

5.3.2 TrackConfiguration

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

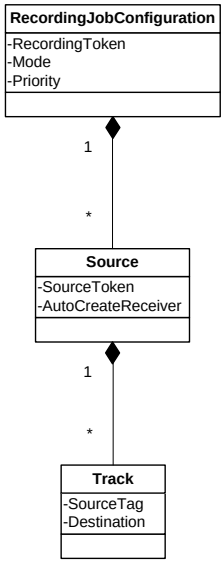
The TrackConfiguration contains the following fields:

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. As a UML diagram, the RecordingJobConfiguration can be viewed as:



The RecordingJobConfiguration holds the following fields:

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

Mode: The mode of the job. If it is idle, nothing shall happen. If it is active, 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 positive 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.

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.

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 tracktoken of the track to which the device shall store the received data.

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.

5.4 CreateRecording

CreateRecording shall create a new recording. The new recording shall be created with one video, one audio and one metadata track.

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

Table 1: CreateRecording command

CreateRecording	Access Class: ACTUATE
Message name	**Description**
CreateRecordingRequest	*Contains the initial configuration for the recording* tt:RecordingConfiguration **RecordingConfiguration**[1][1]
CreateRecordingResponse	*Returns the reference to the created recroding* tt:RecordingReference **RecordingToken**[1][1]
Fault codes	**Description**
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 optinal method is not implemented*

When successfully completed, CreateRecording shall have created three tracks with the following configurations:

TrackToken	**TrackType**
VIDEO001	Video
AUDIO001	Audio
META001	Metadata

All TrackConfigurations shall have the MaximumRetentionTime set to 0 (unlimited), and 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.

Table 2: DeleteRecording command

DeleteRecording	Access Class: ACTUATE
Message name	**Description**
DeleteRecordingRequest	*Identifies the recording that shall be deleted* tt:RecordingReference **RecordingToken**[1][1]
DeleteRecordingResponse	*This message shall be empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoRecording	*The RecordingToken does not reference an exiting recording*
env:Receiver ter: ActionNotSupported ter:NotImplemented	*The device cannot delete recordings*
env:Receiver ter:Action ter:CannotDelete	*This specific recording cannot be deleted*

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.

Table 3: GetRecordings command

GetRecordings	Access Class: READ_MEDIA
Message name	**Description**
GetRecordingsRequest	*This shall be an empty message*
GetRecordingsResponse	*The **RecordingItem** identifies a recording and its current configuration* tt:GetRecordingsResponseItem **RecordingItem**[0][unbounded]
Fault codes	**Description**
No command specific faults	

5.7 SetRecordingConfiguration

SetRecordingConfiguration shall change the configuration of a recording

Table 4: SetRecordingConfiguration command

SetRecordingConfiguration	Access Class: ACTUATE
Message name	**Description**
SetRecordingConfigurationRequest	*The **RecordingToken** shall identify the recording that shall be changed. The* **RecordingConfiguration** *shall be the new configuration for that recording* tt:RecordingReference **RecordingToken**[1][1] tt:RecordingConfiguration **RecordingConfiguration**[1][1]
SetRecordingConfigurationResponse	This message shall be empty.
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter: BadConfiguration	*The configuration is invalid.*
env:Sender ter:InvalidArgVal ter:NoRecording	*The RecordingToken does not reference an existing recording*

5.8 GetRecordingConfiguration

GetRecordingConfiguration shall retrieve the recording configuration for a recording

Table 5: GetRecordingConfiguration command

GetRecordingConfiguration	Access Class: READ_MEDIA
Message name	**Description**
GetRecordingConfigurationRequest	*The **RecordingToken** shall identify the recording for which the configuration shall be retrieved.* tt:RecordingReference **RecordingToken**[1][1]
GetRecordingConfigurationResponse	*The* **RecordingConfiguration** *shall be the current configuration for the specified recording* tt:RecordingConfiguration **RecordingConfiguration**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoRecording	*The RecordingToken does not reference an existing recording*

5.9 CreateTrack

This method shall create a new track within a recording.

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

Table 6: CreateTrack command

CreateTrack	Access Class: ACTUATE
Message name	**Description**
CreateTrackRequest	*The **RecordingToken** shall identify the recording to which a track shall be added. The **TrackConfiguration** shall provide the configuration for the new track.* tt:RecordingReference **RecordingToken**[1][1] tt:TrackConfiguration **TrackConfiguration**[1][1]
CreateTrackResponse	*The **TrackToken** shall identify the newly created track. The **TrackToken** shall be unique within the recoding to which the new track belongs.* tt:TrackReference **TrackToken**[1][1]
Fault codes	**Description**
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 optinal method is not implemented*

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.

Table 7: DeleteTrack command

DeleteTrack	Access Class: ACTUATE
Message name	**Description**
DeleteTrackRequest	*The **RecordingToken** shall identify the recording from which to delete the track. The **TrackToken** identifies the track to delete.* tt:RecordingReference **RecordingToken**[1][1] tt:TrackReference **TrackToken**[1][1]
DeleteTrackResponse	*This message shall be empty.*

Fault codes	Description
env:Receiver ter:ActionNotSupported ter:NotImplemented	*The device does not implement the DeleteTrack method.*
env:Sender ter:InvalidArgVal ter:NoTrack	*The TrackToken does not reference an existing track of the recording.*
env:Sender ter:InvalidArgVal ter:NoRecording	*The RecordingToken does not reference an existing recording*
env:Receiver ter:Action ter:CannotDelete	*This specific track cannot be deleted*

5.11 GetTrackConfiguration

GetTrackConfiguration shall retrieve the configuration for a specific track.

Table 8: GetTrackConfiguration command

GetTrackConfiguration	Access Class: READ_MEDIA
Message name	**Description**
GetTrackConfigurationRequest	*The **RecordingToken** and **TrackToken** shall identify the recording from which to get the track configuration.* tt:RecordingReference **RecordingToken**[1][1] tt:TrackReference **TrackToken**[1][1]
GetTrackConfigurationResponse	tt:TrackConfiguration **TrackConfiguration**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoTrack	*The TrackToken does not reference an existing track of the recording.*
env:Sender ter:InvalidArgVal ter:NoRecording	*The RecordingToken does not reference an existing recording*

5.12 SetTrackConfiguration

SetTrackConfiguration shall change the configuration of a track.

Table 9: SetTrackConfiguration command

SetTrackConfiguration	Access Class: ACTUATE
Message name	**Description**
SetTrackConfigurationRequest	*The **RecordingToken** and **TrackToken** shall identify the track for which to set the track configuration. The **TrackConfiguration** is the new configuration for the track.* tt:RecordingReference **RecordingToken**[1][1] tt:TrackReference **TrackToken**[1][1] tt:TrackConfiguration **TrackConfiguration**[1][1]

SetTrackConfigurationResponse	This message shall be empty.
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoTrack	*The TrackToken does not reference an existing track of the recording.*
env:Sender ter:InvalidArgVal ter:NoRecording	*The RecordingToken does not reference an existing recording*
env:Sender ter:InvalidArgVal ter:BadConfiguration	*The contents of the configuration object are invalid.*

5.13 CreateRecordingJob

CreateRecordingJob shall create a new recording job.

Table 10: CreateRecordingJob command

CreateRecordingJob	Access Class: ACTUATE
Message name	**Description**
CreateRecordingJobRequest	***JobConfiguration*** *shall hold the configuration for the new recording job.* tt:RecordingJobConfiguration **JobConfiguration**[1][1]
CreateRecordingJobResponse	*The* ***JobToken*** *shall identify the created recording job. The* ***JobConfiguration*** *structure shall be the configuration as it is used by the device. This may be different from the JobConfiguration passed to CreateRecordingJob.* tt:RecordingJobReference **JobToken**[1][1] tt:RecordingJobConfiguration **JobConfiguration**[1][1]
Fault codes	**Description**
env:Receiver ter:Action ter:MaxRecordingJobs	*The maximum number of recording jobs that the device can handle has been reached.*
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.*

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.

Table 11: DeleteRecordingJob command

DeleteRecordingJob	Access Class: ACTUATE
Message name	**Description**
DeleteRecordingJobRequest	*The **JobToken** shall identify the recording job that shall be deleted.* tt:RecordingJobReference **JobToken**[1][1]
DeleteRecordingJobResponse	*The message shall be empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoRecordingJob	*The JobToken does not reference an exiting job*

5.15 GetRecordingJobs

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

Table 12: GetRecordingJobs command

GetRecordingJobs	Access Class: READ_MEDIA
Message name	**Description**
GetRecordingJobsRequest	*This message shall be empty.*
GetRecordingJobsResponse	*The **JobItem** identifies a job in the device and holds its current configuration.* tt:GetRecordingJobsResponseItem **JobItem**[0][unbounded]
Fault codes	**Description**
No command specific faults	

5.16 SetRecordingJobConfiguration

SetRecordingJobConfiguration shall change the configuration for a recording job.

Table 13: SetRecordingJobConfiguration command

SetRecordingJobConfiguration	Access Class: ACTUATE
Message name	**Description**
SetRecordingJobConfigurationRequest	*The **JobConfiguration** returned from SetRecordingJobConfiguration 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. tt:RecordingJobReference **JobToken**[1][1] tt:RecordingJobConfiguration **JobConfiguration**[1][1]
SetRecordingJobConfigurationResponse	*The **JobConfiguration** structure shall be the configuration as it is used by the device. This may be different from the **JobConfiguration** passed to CreateRecordingJob.* tt:RecordingJobConfiguration **JobConfiguration**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoRecordingJob	*The JobToken does not reference an exiting 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.*

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.

Table 14: GetRecordingJobConfiguration command

GetRecordingJobConfiguration	Access Class: READ_MEDIA
Message name	**Description**
GetRecordingJobConfiguration Request	*The **JobToken** shall identify the recording job for which to retrieve the configuration.* tt:RecordingJobReference **JobToken**[1][1]
GetRecordingJobConfiguration Response	*The **JobConfiguration** shall hold the current configuration of the recording job.* tt:RecordingJobConfiguration **JobConfiguration**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoRecordingJob	*The JobToken does not reference an exiting job*

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.

Table 15: SetRecordingJobMode command

SetRecordingJobMode	Access Class: ACTUATE
Message name	**Description**
SetRecordingJobModeRequest	*The **JobToken** shall identify the recording job for which to change the recording mode. The **Mode** shall be the new mode for the recording job.* tt:RecordingJobReference **JobToken**[1][1] tt:RecordingJobMode **Mode**[1][1]
SetRecordingJobModeResponse	*This message shall be empty.*
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoRecordingJob	*The JobToken does not reference an exiting job*
env:Sender ter:InvalidArgVal ter:BadMode	*The Mode is invalid.*

5.19 GetRecordingJobState

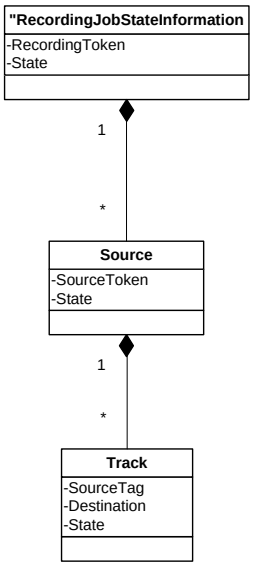
GetRecordingJobState returns the state of a recording job. It includes an aggregated state, and state for each track of the recording job. The RecordingJogState may change due to

- 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.

Table 16: GetRecordingJobState command

GetRecordingJobState	Access Class: READ_MEDIA
Message name	**Description**
GetRecordingJobState Request	*The **JobToken** shall identify the recording job for which to get the state.* tt:RecordingJobReference **JobToken**[1][1]
GetRecordingJobState Response	*The **State** shall hold the state of the recording job.* tt:RecordingJobStateInformation **State**[1][1]
Fault codes	**Description**
env:Sender ter:InvalidArgVal ter:NoRecordingJob	*The JobToken does not reference an exiting job*

The UML representation of the RecordingJobStateInformation structure is:



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

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 Capabilities

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

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

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

DeleteData Indication if the device supports explicit deletion of data.

Encoding Indication which encodings are supported for recording. The list may contain one or more enumeration values of tt:VideoEncoding and tt:AudioEncoding.

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.

Table 17: GetServiceCapabilities command

GetServiceCapabilities	Access Class: PRE_AUTH
Message name	**Description**
GetServiceCapabilitiesRequest	*This is an empty message.*
GetServiceCapabilitiesResponse	*The capability response message contains the requested service capabilities using a hierarchical XML capability structure.* trc:Capabilities **Capabilities** [1][1]
Fault codes	**Description**
	No command specific faults!

5.21 Events

The recording service shall dispatch events through the event service. It shall be capable of generating the events listed in this chapter whenever the condition that fires the event occurs.

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.

5.21.1 Recording job state changes

If the a state field of the RecordingJobStateInformation structure changes, the device shall send the 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>
```

5.21.2 Configuration changes

If the configuration of a recording is changed, the device shall send the 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, the device shall send the 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, the device shall send the 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>
```

5.21.3 Data deletion

Whenever data is deleted, the device shall send the 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>
```

5.21.4 Recording and track creation and deletion

Whenever a recording is created, the device shall send the 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, the device shall send the 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, the device shall send the 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, the device shall send the 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>
```

5.22 Examples

5.22.1 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.

5.22.2 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)
```

5.23 Service specific data types

5.23.1 RecordingInformation

```
<xs:complexType name="RecordingInformation">
   <xs:element name="RecordingToken" type="tt:RecordingReference"/>
   <xs:element name="Source" type="tt:RecordingSourceInformation/>
   <xs:element name="EarliestRecording" type="xs:dateTime" minOccurs="0"/>
   <xs:element name="LatestRecording" type="xs:dateTime" minOccurs="0"/>
   <xs:element name="Content" type="tt:Description"/>
```

```
   <xs:element name="Track" type="tt:TrackInformation minOccurs="0"
            maxOccurs="unbounded"/>
   <xs:element name="RecordingStatus" type="tt:RecordingStatus"/>
</xs:complexType>
```

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

- **Source**
  Information about the source of the recording. This gives a description of where the data in the recording comes from. Since a single recording is intended to record related material, there is just one source. It is indicates the physical location or the major data source for the recording. Currently the recordingconfiguration cannot describe each individual data source.

- **EarliestRecording**
  the date and time of the oldest data in the recording

- **LatestRecording**
  the date and time of the newest data in the recording.

- **Content**
  informative description of content.

- **Track**
  Contains information about a single track in a recording.

- **RecordingStatus**
  current status of recording, can be any of: Initiated, Recording, Stopped, Removing, Removed.

5.23.2 RecordingSourceInformation

A set of informative desciptions of a data source. The Search searvice allows a client to filter on recordings based on information in this structure.

```
<xs:complexType name="RecordingSourceInformation">
   <xs:element name="SourceId" type="xs:anyURI"/>
   <xs:element name="Name" type="tt:Name/>
   <xs:element name="Location" type="tt:Description"/>
   <xs:element name="Description" type="tt:Description"/>
   <xs:element name="Address" type="xs:anyURI"/>
</xs:complexType>
```

- **SourceId**
  Identifier for the source chosen by the client that creates the structure. This identifier is opaque to the device. Clients may use any type of URI for this field.

- **Name**
  Informative user readable name of the source, e.g. "Camera23".

- **Location**
  Informative description of the physical location of the source, e.g. the coordinates on a map.

- **Description**
  Informative description of the source.

- **Address**
  URI provided by the service supplying data to be recorded.

5.23.3 TrackInformation

```
<xs:complexType name="TrackInformation">
   <xs:element name="TrackToken" type="tt:TrackReference"/>
   <xs:element name="TrackType" type="tt:TrackType"/>
```

```
   <xs:element name="Description" type="tt:Description"/>
   <xs:element name="DataFrom" type="xs:dateTime"/>
   <xs:element name="DataTo" type="xs:dateTime"/>
</xs:complexType>
```

- **TrackToken**
  an identifier of the track.

- **TrackType**
  Type of the track: "Video", "Audio" or "Metadata". The track shall only be able to hold data of that type.

- **Description**
  Informative description of the contents of the track.

- **DataFrom**
  The date and time of the oldest data in the track.

- **DataTo**
  The date and time of the newest data in the track.

5.23.4 MediaAttributes

A set of media attributes valid for a recording at a point in time or for a time interval.

```
<xs:complexType name="MediaAttributes">
   <xs:element name="RecordingToken" type="tt:RecordingReference"/>
   <xs:element name="TrackAttributes" type="tt:TrackAttributes
            minOccurs="0" maxOccurs="unbounded"/>
   <xs:element name="From" type="xs:dateTime"/>
   <xs:element name="Until" type="xs:dateTime"/>
</xs:complexType>
```

- **RecordingToken**
  A reference to the recording that has these attributes.

- **TrackAttributes**
  A set of attributes for each track.

- **From**
  The attributes are valid from this point in time in the recording.

- **Until**
  The attributes are valid until this point in time in the recording. Can be equal to 'From' to indicate that the attributes are only known to be valid for this particular point in time.

5.23.5 TrackAttributes

```
<xs:complexType name="TrackAttributes">
   <xs:element name="TrackInformation" type="tt:TrackInformation/>
   <xs:element name="VideoAttributes" type="tt:VideoAttributes
            minOccurs="0"/>
   <xs:element name="AudioAttributes" type="tt:AudioAttributes
            minOccurs="0"/>
   <xs:element name="MetadataAttributes" type="tt:MetadataAttributes
            minOccurs="0"/>
</xs:complexType>
```

- **TrackInformation**
  The basic information about the track.

- **VideoAttributes**
  If the track is a video track, exactly one of this structure shall be present and contain the video attributes.

- **AudioAttributes**
  If the track is an audio track, exactly one of this structure shall be present and contain the audio attributes.

- **MetadataAttributes**
  If the track is an metadata track, exactly one of this structure shall be present and contain the metadata attributes.

5.23.6 VideoAttributes

```
<xs:complexType name="VideoAttributes">
   <xs:element name="Bitrate" type="xs:int" minOccurs="0"/>
   <xs:element name="Width" type="xs:int"/>
   <xs:element name="Height" type="xs:int"/>
   <xs:element name="Encoding" type="tt:VideoEncoding"/>
   <xs:element name="Framerate" type="xs:float"/>
</xs:complexType>
```

- **Bitrate**
  Average bitrate in kbps.

- **Width**
  The width of the video in pixels.

- **Height**
  The height of the video in pixels.

- **Encoding**
  Used video codec, either Jpeg, H.264 or Mpeg4

- **Framerate**
  Average framerate in frames per second.

5.23.7 AudioAttributes

```
<xs:complexType name="AudioAttributes">
   <xs:element name="Bitrate" type="xs:int" minOccurs="0"/>
   <xs:element name="Encoding" type="tt:AudioEncoding"/>
   <xs:element name="Samplerate" type="xs:int"/>
</xs:complexType>
```

- **Bitrate**
  The bitrate in kbps.

- **Encoding**
  Audio codec used for encoding the audio (either G.711, G.726 or AAC)

- **Samplerate**
  The sample rate in kHz.

5.23.8 MetadataAttributes

```
<xs:complexType name="MetadataAttributes">
   <xs:element name="CanContainPTZ" type="xs:boolean"/>
   <xs:element name="CanContainAnalytics" type="xs:boolean"/>
   <xs:element name="CanContainNotifications" type="xs:boolean"/>
</xs:complexType>
```

- **CanContainPTZ**
  Indicates that there can be PTZ data in the metadata track in the specified time interval.

- **CanContainAnalytics**
  Indicates that there can be analytics data in the metadata track in the specified time interval.

- **CanContainNotifications**
  Indicates that there can be notifications in the metadata track in the specified time interval.

5.23.9 RecordingConfiguration

```
<xs:complexType name="RecordingConfiguration">
   <xs:element name="Source" type="tt:RecordingSourceInformation/>
   <xs:element name="Content" type="tt:Description"/>
```

```
   <xs:element name="MaximumRetentionTime" type="xs:duration"/>
</xs:complexType>
```

- **Source**
  Information about the source of the recording.

- **Content**
  Informative description of the source.

- **MaximumRetentionTime**
  Sspecifies 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.

5.23.10 TrackConfiguration

```
<xs:complexType name="TrackConfiguration">
   <xs:element name="TrackType" type="tt:TrackType"/>
   <xs:element name="Description" type="tt:Description"/>
</xs:complexType>
```

- **TrackType**
  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.

- **Description**
  Informative description of the track.

5.23.11 GetRecordingsResponseItem

```
<xs:complexType name="GetRecordingsResponseItem">
   <xs:element name="RecordingToken" type="tt:RecordingReference"/>
   <xs:element name="Configuration" type="tt:RecordingConfiguration/>
   <xs:element name="Tracks" type="tt:GetTracksResponseList/>
</xs:complexType>
```

- **RecordingToken**
  Token of the recording.

- **Configuration**
  Configuration of the recording.

- **Tracks**
  List of tracks.

5.23.12 GetTracksResponseList

```
<xs:complexType name="GetTracksResponseList">
   <xs:element name="Track" type="tt:GetTracksResponseItem minOccurs="0"
            maxOccurs="unbounded"/>
</xs:complexType>
```

- **Track**
  Configuration of a track.

5.23.13 GetTracksResponseItem

```
<xs:complexType name="GetTracksResponseItem">
   <xs:element name="TrackToken" type="tt:TrackReference"/>
   <xs:element name="Configuration" type="tt:TrackConfiguration/>
</xs:complexType>
```

- **TrackToken**
  Token of the track.

- **Configuration**
  Configuration of the track.

5.23.14 RecordingJobConfiguration

```
<xs:complexType name="RecordingJobConfiguration">
   <xs:element name="RecordingToken" type="tt:RecordingReference"/>
   <xs:element name="Mode" type="tt:RecordingJobMode"/>
   <xs:element name="Priority" type="xs:int"/>
   <xs:element name="Source" type="tt:RecordingJobSource minOccurs="0"
            maxOccurs="unbounded"/>
</xs:complexType>
```

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

- **Mode**
  The mode of the job. If it is idle, nothing shall happen. If it is active, 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 positive 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.

- **Source**
  Source of the recording.

5.23.15 RecordingJobSource

```
<xs:complexType name="RecordingJobSource">
   <xs:element name="SourceToken" type="tt:SourceReference minOccurs="0"/>
   <xs:element name="AutoCreateReceiver" type="xs:boolean" minOccurs="0"/>
   <xs:element name="Tracks" type="tt:RecordingJobTrack minOccurs="0"
            maxOccurs="unbounded"/>
</xs:complexType>
```

- **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.

- **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.

- **Tracks**
  List of tracks associated with the recording.

5.23.16 RecordingJobTrack

```
<xs:complexType name="RecordingJobTrack">
   <xs:element name="SourceTag" type="xs:string"/>
   <xs:element name="Destination" type="tt:TrackReference"/>
</xs:complexType>
```

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

- **Destination**
  The destination is the tracktoken of the track to which the device shall store the received data.

5.23.17 RecordingJobStateInformation

```
<xs:complexType name="RecordingJobStateInformation">
   <xs:element name="RecordingToken" type="tt:RecordingReference"/>
   <xs:element name="State" type="tt:RecordingJobState"/>
   <xs:element name="Sources" type="tt:RecordingJobStateSource
            minOccurs="0" maxOccurs="unbounded"/>
</xs:complexType>
```

- **RecordingToken**
  Identification of the recording that the recording job records to.

- **State**
  Holds the aggregated state over the whole RecordingJobInformation structure.

- **Sources**
  Identifies the data source of the recording job.

5.23.18 RecordingJobStateSource

```
<xs:complexType name="RecordingJobStateSource">
   <xs:element name="SourceToken" type= "tt:SourceReference/>
   <xs:element name="State" type="tt:RecordingJobState"/>
   <xs:element name="Tracks" type="tt:RecordingJobStateTracks/>
</xs:complexType>
```

- **SourceToken**
  Identifies the data source of the recording job.

- **State**
  Holds the aggregated state over all substructures of RecordingJobStateSource.

- **Tracks**
  List of track items.

5.23.19 RecordingJobStateTracks

```
<xs:complexType name="RecordingJobStateTracks">
   <xs:element name="Track" type="tt:RecordingJobStateTrack minOccurs="0"
            maxOccurs="unbounded"/>
</xs:complexType>
```

- **Track**
  an identifier of the track.

5.23.20 RecordingJobStateTrack

```
<xs:complexType name="RecordingJobStateTrack">
   <xs:element name="SourceTag" type="xs:string"/>
   <xs:element name="Destination" type="tt:TrackReference"/>
   <xs:element name="Error" type="xs:string" minOccurs="0"/>
   <xs:element name="State" type="tt:RecordingJobState"/>
</xs:complexType>
```

- **SourceTag**
  Identifies the track of the data source that provides the data.

- **Destination**
  Indicates the destination track.

- **Error**
  Optionally holds an implementation defined string value that describes the error. The string should be in the English language.

- **State**
  Provides 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.

5.23.21 GetRecordingJobsResponseItem

```
<xs:complexType name="GetRecordingJobsResponseItem">
   <xs:element name="JobToken" type="tt:RecordingJobReference"/>
   <xs:element name="JobConfiguration" type="tt:RecordingJobConfiguration"/>
</xs:complexType>
```

- **JobToken**
  identifier of a job.

- **JobConfiguration**
  holds the configuration for a recording job

Annex A. 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