

ONVIF™
Recording Search
Service Specification

Version 2.3
May, 2013



© 2008-2013 by ONVIF: Open Network Video Interface Forum Inc.. All rights reserved.

Recipients of this document may copy, distribute, publish, or display this document so long as this copyright notice, license and disclaimer are retained with all copies of the document. No license is granted to modify this document.

THIS DOCUMENT IS PROVIDED "AS IS," AND THE CORPORATION AND ITS MEMBERS AND THEIR AFFILIATES, MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THIS DOCUMENT ARE SUITABLE FOR ANY PURPOSE; OR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

IN NO EVENT WILL THE CORPORATION OR ITS MEMBERS OR THEIR AFFILIATES BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE OR CONSEQUENTIAL DAMAGES, ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THIS DOCUMENT, WHETHER OR NOT (1) THE CORPORATION, MEMBERS OR THEIR AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR (2) SUCH DAMAGES WERE REASONABLY FORESEEABLE, AND ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THIS DOCUMENT. THE FOREGOING DISCLAIMER AND LIMITATION ON LIABILITY DO NOT APPLY TO, INVALIDATE, OR LIMIT REPRESENTATIONS AND WARRANTIES MADE BY THE MEMBERS AND THEIR RESPECTIVE AFFILIATES TO THE CORPORATION AND OTHER MEMBERS IN CERTAIN WRITTEN POLICIES OF THE CORPORATION.

CONTENTS

- 1 Scope** **5**
- 2 Normative references** **5**
- 3 Terms and Definitions** **5**
 - 3.1 Definitions.....5
 - 3.2 Abbreviations.....5
- 4 Overview** **6**
- 5 Service** **6**
 - 5.1 Introduction.....6
 - 5.2 Concepts7
 - 5.2.1 Search Direction7
 - 5.2.2 Recording Event7
 - 5.2.3 Search Session.....8
 - 5.2.4 Search Scope8
 - 5.2.5 Search Filters.....8
 - 5.2.6 Time Information.....8
 - 5.3 Data Structures9
 - 5.3.1 RecordingInformation Structure.....9
 - 5.3.2 RecordingSourceInformation Structure10
 - 5.3.3 TrackInformation Structure10
 - 5.3.4 SearchState Enumeration.....10
 - 5.3.5 MediaAttributes Structure10
 - 5.3.6 FindEventResult Structure.....11
 - 5.3.7 FindPTZPositionResult Structure11
 - 5.3.8 PTZPositionFilter Structure.....11
 - 5.3.9 MetadataFilter Structure11
 - 5.3.10 FindMetadataResult Structure.....11
 - 5.4 GetRecordingSummary.....12
 - 5.5 GetRecordingInformation12
 - 5.6 GetMediaAttributes13
 - 5.7 FindRecordings13
 - 5.8 GetRecordingSearchResults.....14
 - 5.9 FindEvents15
 - 5.10 GetEventSearchResults.....16
 - 5.11 FindPTZPosition.....17
 - 5.12 GetPTZPositionSearchResults18
 - 5.13 FindMetadata19
 - 5.14 GetMetadataSearchResults20
 - 5.15 GetSearchState.....21
 - 5.16 EndSearch.....21
 - 5.17 Capabilities.....22
 - 5.18 Service specific fault codes.....22
 - 5.19 Recording Event Descriptions.....23
 - 5.20 XPath dialect.....24
 - 5.21 Service specific data types.....25

5.21.1	SearchScope	25
5.21.2	EventFilter	25
5.21.3	PTZPositionFilter	25
5.21.4	MetadataFilter	26
5.21.5	FindRecordingResultList	26
5.21.6	FindEventResultList	26
5.21.7	FindEventResult	26
5.21.8	FindPTZPositionResultList	27
5.21.9	FindPTZPositionResult	27
5.21.10	FindMetadataResultList	27
5.21.11	FindMetadataResult	28

Annex A. Revision History**29**

1 Scope

This document defines the web service interface for searching for recorded Video, Audio and Metadata.

For a definition of the storage model see the ONVIF Recording Control Specification.

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>>

ONVIF Recording Control Specification

<<http://www.onvif.org/specs/srv/rec/ONVIF-RecordingControl-Service-Spec-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

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

The search service provides the following functionality:

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

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

FindRecordings and GetRecordingSearchResults

FindEvents and GetEventSearchResults

FindPTZPosition and GetPTZPositionSearchResults

FindMetadata and GetMetadataSearchResults

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

5 Service

5.1 Introduction

The search service provides a number of operations for finding data of interest within a set of recordings. The most common way of doing this would be to search for events that are either

included in the metadata track of a recording, or are otherwise associated with a recording in the device (see Recording Events below).

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

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

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

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

GetSearchState returns the state of a search session.

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

5.2 Concepts

5.2.1 Search Direction

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

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

5.2.2 Recording Event

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

However created, recording events are always treated as notifications in regards to search filters and results returned. Each recording event has a notification topic as defined in the Topic Structure section of the ONVIF Core Specification. Predefined recording events are described in section 5.18.

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

5.2.3 Search Session

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

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

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

5.2.4 Search Scope

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

5.2.4.1 Included data

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

5.2.4.2 Recording Information Filter

Rather than specifying a list of recording tokens, the recordings can be filtered by an XPath filter operating on the RecordingInformation structure. This allows the client to filter on all elements present in the RecordingInformation structure, using comparisons according to the XPath dialect defined in section 5.19. If a recording information filter is supplied, only recordings matching the filter shall be part of the scope.

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

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

5.2.5 Search Filters

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

5.2.6 Time Information

An ONVIF compliant device shall support time values in request parameters that are given in utc with the 'Z' indicator and respond all time values as utc including the 'Z' indicator.

5.3 Data Structures

5.3.1 RecordingInformation Structure

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

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

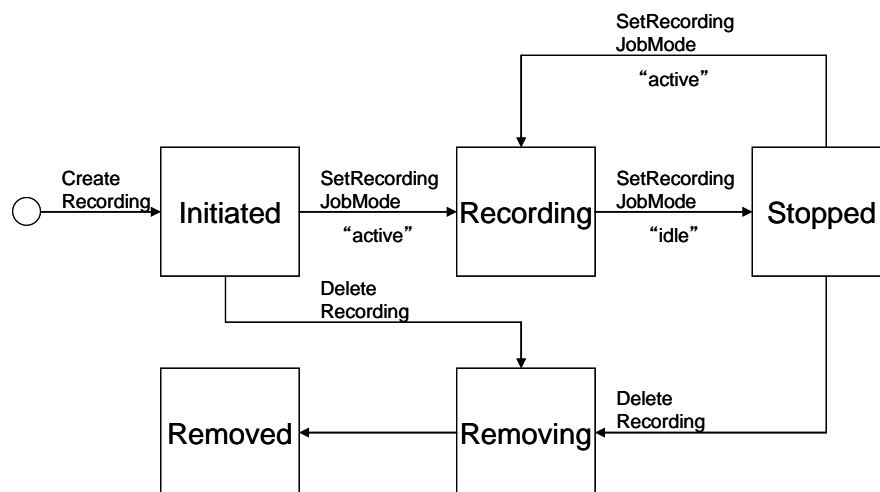


Figure 1: Recording state chart.

Figure 1 shows the state changes for the RecordingStatus. The following adds additional explanation:

- Initiated – the new recording is created and the specified recording has not yet started to record.
- Recording – the recording job is executing the specified recording.
- Stopped – the recording job is pausing the specified recording. This case requires that the job has started recording before at least once.
- Removing :The specified recording is in the process of being deleted.
- Removed :The specified recording has been removed.

- Unknown : This case should never happen.

5.3.2 RecordingSourceInformation Structure

Contains information about the source of a recording.

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

5.3.3 TrackInformation Structure

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

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

5.3.4 SearchState Enumeration

The search state can be one of the following

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

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

5.3.5 MediaAttributes Structure

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

- RecordingToken – A reference to the recording that this structure concerns.

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

5.3.6 FindEventResult Structure

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

5.3.7 FindPTZPositionResult Structure

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

5.3.8 PTZPositionFilter Structure

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

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

5.3.9 MetadataFilter Structure

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

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

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

5.3.10 FindMetadataResult Structure

- RecordingToken – identifying the recording containing the matching metadata.

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

5.4 GetRecordingSummary

GetRecordingSummary is used to get a summary description of all recorded data. This operation is mandatory to support for a device implementing the recording search service.

Table 2: GetRecordingSummary command

GetRecordingSummary		Access Class: READ_MEDIA
Message name	Description	
GetRecordingSummaryRequest	This shall be the empty message	
GetRecordingSummaryResponse	Returns a structure containing: <i>DataFrom</i> specifying the first time when there is recorded data on the device; <i>DataUntil</i> specifying the last point in time where there is data recorded on the device; the total number of recordings on the device. tt:RecordingSummary Summary [1][1]	
Fault codes	Description	
	<i>No command specific error codes.</i>	

5.5 GetRecordingInformation

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

Table 3: GetRecordingInformation command

GetRecordingInformation		Access Class: READ_MEDIA
Message name	Description	
GetRecordingInformationRequest	<i>Request description</i> tt:ReferenceToken RecordingToken [1][1]	
GetRecordingInformationResponse	<i>Response description</i> tt:RecordingInformation RecordingInformation [1][1]	
Fault codes	Description	
env:Sender ter:InvalidArgVal ter:InvalidToken	<i>The RecordingToken is not valid.</i>	

5.6 GetMediaAttributes

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

Table 4: GetMediaAttributes command

GetMediaAttributes		Access Class: READ_MEDIA
Message name	Description	
GetMediaAttributesRequest	<i>RecordingTokens is a list of references to the recordings to query. Time is the point in time where for which information is requested.</i> tt:ReferenceToken RecordingTokens [0][unbounded] xs:dateTime Time [1][1]	
GetMediaAttributesResponse	<i>Contains a MediaAttributes structure for the RecordingToken specified in the request. Note that each RecordingToken can result in zero or one MediaAttributes.</i> tt:MediaAttributes MediaAttributes [0][unbounded]	
Fault codes	Description	
env:Sender ter:InvalidArgVal ter:InvalidToken	<i>The RecordingToken is not valid.</i>	
env:Sender ter:InvalidArgVal ter:NoRecording	<i>No recording available.</i>	

5.7 FindRecordings

FindRecordings starts a search session, looking for recordings that matches the scope (See 5.2.4) defined in the request. Results from the search are acquired using the GetRecordingSearchResults request, specifying the search token returned from this request.

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

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

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

For the KeepAliveTime a device shall support at least values up to ten seconds. A device may adapt larger values.

Table 5: FindRecordings command

FindRecordings		Access Class: READ_MEDIA
Message name	Description	
FindRecordingsRequest	<p><i>Scope defines the dataset to consider for this search. The search ends after MaxMatches. KeepAliveTime is the session timeout after each request concerning this session.</i></p> <p>tt:SearchScope Scope[1][1] xs:int MaxMatches[0][1] xs:duration KeepAliveTime[1][1]</p>	
FindRecordingsResponse	<p><i>Returns the SearchToken identifying the search session created by this request.</i></p> <p>tt:JobToken SearchToken[1][1]</p>	
Fault codes	Description	
env:Receiver ter:Action ter:ResourceProblem	<i>Device is unable to create a new search session.</i>	
env:Sender ter:InvalidArgVal ter:InvalidToken	<i>The recording token is invalid.</i>	
env:Sender ter:InvalidArgVal ter:InvalidSource	<i>The recording source is invalid.</i>	

5.8 GetRecordingSearchResults

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

GetRecordingSearchResults shall block until:

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

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

Table 6: GetRecordingSearchResults command

GetRecordingSearchResults		Access Class: READ_MEDIA
Message name	Description	
GetRecordingSearchResults Request	<p><i>SearchToken</i> specifies the search session. <i>MinResults</i> specifies the minimum number of results that should be returned. If the total number of results is lower than <i>MinResults</i> in a completed search, all results should be returned. <i>MaxResults</i> specifies the maximum number of results to return. <i>WaitTime</i> defines the maximum time to block, waiting for results.</p> <p>tt:JobToken SearchToken [1][1] xs:int MinResults [0][1] xs:int MaxResults [0][1] xs:duration WaitTime [0][1]</p>	
GetRecordingSearchResults Response	<p>Returns a structure containing the current <i>SearchState</i> and a list of <i>RecordingInformation</i> structures.</p> <p>tt:FindRecordingResultList ResultList[1][1]</p>	
Fault codes	Description	
env:Sender ter:InvalidArgVal ter:InvalidToken	<p>The search token is invalid.</p>	

5.9 FindEvents

FindEvents starts a search session, looking for events in the *scope* (See 5.2.4) that match the search filter defined in the request. Events are recording events (see 5.2.2) and other events that are available in the track. Results from the search are acquired using the GetEventSearchResults request, specifying the search token returned from this request.

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

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

Results shall be ordered by time, ascending in case of forward search, or descending in case of backward search. This operation is mandatory to support for a device implementing the recording search service. Although the values of property events refer to the forward direction, they shall be reported identically in reverse search mode.

For the KeepAliveTime a device shall support at least values up to ten seconds. A device may adapt larger values.

Table 7: FindEvents command

FindEvents		Access Class: READ_MEDIA
Message name	Description	
FindEventsRequest	<p><i>StartPoint</i> is the point of time where the search will start. <i>EndPoint</i> is the point of time where the search will stop. This can be a time before the <i>StartPoint</i>, in which case the search is performed backwards in time. If <i>EndPoint</i> is omitted, search will go forward from the <i>StartPoint</i>. <i>Scope</i> defines the dataset to consider for this search. <i>SearchFilter</i> contains the topic and message filter needed to define what events to search for. By setting the <i>IncludeStartState</i> to true, the client indicates that virtual events at the time of <i>StartPoint</i> should be returned to represent the state in the recording. In case of a backward search, virtual events at the time of <i>EndPoint</i> and <i>StartPoint</i> should be returned. Support for virtual events is mandatory for recording events. Support for additional virtual events is signaled via the <i>GeneralStartEvents</i> capability. The search ends after <i>MaxMatches</i>. <i>KeepAliveTime</i> is the session timeout after each request concerning this session.</p> <p>xs:dateTime StartPoint[1][1] xs:dateTime EndPoint[0][1] tt:SearchScope Scope[1][1] tt:EventFilter SearchFilter[1][1] xs:boolean IncludeStartState[1][1] xs:int MaxMatches[0][1] xs:duration KeepAliveTime[1][1]</p>	
FindEventsResponse	<p>Returns the <i>SearchToken</i> identifying the search session created by this request.</p> <p>tt:JobToken SearchToken[1][1]</p>	
Fault codes	Description	
env:Receiver ter:Action ter:ResourceProblem	<p>Device is unable to create a new search session.</p>	

5.10 GetEventSearchResults

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

GetEventSearchResults shall block until:

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

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

Table 8: GetEventSearchResults command

GetEventSearchResults		Access Class: READ_MEDIA
Message name	Description	
GetEventSearchResultsRequest	<p><i>SearchToken</i> specifies the search session. <i>MinResults</i> specifies the minimum number of results that should be returned. <i>MaxResults</i> specifies the maximum number of results to return. <i>WaitTime</i> defines the maximum time to block, waiting for results.</p> <p>tt:JobToken SearchToken [1][1] xs:int MinResults [0][1] xs:int MaxResults [0][1] xs:duration WaitTime [0][1]</p>	
GetEventSearchResultsResponse	<p>Returns a structure containing the current <i>SearchState</i> and a list of <i>FindEventResult</i> structures.</p> <p>tt:SearchState SearchState [1][1] tt:FindEventResult FindEventResult [0][unbounded]</p>	
Fault codes	Description	
env:Sender ter:InvalidArgVal ter:InvalidToken	<p><i>The search token is invalid.</i></p>	

5.11 FindPTZPosition

FindPTZPosition starts a search session, looking for ptz positions in the *scope* (See 5.2.4) that matches the search filter defined in the request. Results from the search are acquired using the `GetPTZPositionSearchResults` request, specifying the search token returned from this request.

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

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

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

For the `KeepAliveTime` a device shall support at least values up to ten seconds. A device may adapt larger values.

Table 9: FindPTZPosition command

FindPTZPosition		Access Class: READ_MEDIA
Message name	Description	
FindPTZPositionRequest	<p><i>StartPoint</i> is the point of time where the search will start. <i>EndPoint</i> is The point of time where the search will stop. This can be a time before the <i>StartPoint</i>, in which case the search is performed backwards in time. If <i>EndPoint</i> is omitted, search will go forward from the <i>StartPoint</i>. <i>Scope</i> defines the dataset to consider for this search. <i>SearchFilter</i> contains the search criteria needed to define the PTZ position to search for. The search ends after <i>MaxMatches</i>. <i>KeepAliveTime</i> is the session timeout after each request concerning this session.</p> <p>xs:dateTime StartPoint[1][1] xs:dateTime EndPoint[0][1] tt:SearchScope Scope[1][1] tt:PTZPositionFilter SearchFilter[1][1] xs:int MaxMatches[0][1] xs:duration KeepAliveTime[1][1]</p>	
FindPTZPositionResponse	<p>Returns the <i>SearchToken</i> identifying the search session created by this request.</p> <p>tt:JobToken SearchToken[1][1]</p>	
Fault codes	Description	
env:Receiver ter:Action ter:ResourceProblem	Device is unable to create a new search session.	
env:Receiver ter:ActionNotSupported ter:NotImplemented	This optional method is not implemented	

5.12 GetPTZPositionSearchResults

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

GetPTZPositionSearchResults shall block until:

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

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

Table 10: GetPTZPositionSearchResults command

GetPTZPositionSearchResults		Access Class: READ_MEDIA
Message name	Description	
GetPTZPositionSearchResult sRequest	<p><i>SearchToken</i> specifies the search session. <i>MinResults</i> specifies the minimum number of results that should be returned. <i>MaxResults</i> specifies the maximum number of results to return. <i>WaitTime</i> defines the maximum time to block, waiting for results.</p> <p>tt:JobToken SearchToken[1][1] xs:int MinResults[0][1] xs:int MaxResults[0][1] xs:duration WaitTime[0][1]</p>	
GetPTZPositionSearchResult sResponse	<p>Returns a structure containing the current <i>SearchState</i> and a list of <i>FindPTZPositionResult</i> structures.</p> <p>tt:SearchState SearchState[1][1] tt:FindPTZPositionResult FindPTZPositionResult[0][unbounded]</p>	
Fault codes	Description	
env:Sender ter:InvalidArgVal ter:InvalidToken	<p>The search token is invalid.</p>	

5.13 FindMetadata

FindMetadata starts a search session, looking for metadata in the scope (See 5.2.4) that matches the search filter defined in the request. Results from the search are acquired using the GetMetadataSearchResults request, specifying the search token returned from this request.

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

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

This operation is mandatory to support if the MetaDataSearch capability is set to true in the SearchCapabilities structure return by the GetCapabilities command in the Device service.

For the KeepAliveTime a device shall support at least values up to ten seconds. A device may adapt larger values.

Table 11: FindMetadata command

FindMetadata		Access Class: READ_MEDIA
Message name	Description	
FindMetadataRequest	<p><i>StartPoint</i> is the point of time where the search will start. <i>EndPoint</i> is the point of time where the search will stop. This can be a time before the <i>StartPoint</i>, in which case the search is performed backwards in time. If <i>EndPoint</i> is omitted, search will go forward from the <i>StartPoint</i>. <i>Scope</i> defines the dataset to consider for this search. <i>SearchFilter</i> contains the search criteria needed to define the metadata to search for. The search ends after <i>MaxMatches</i>. <i>KeepAliveTime</i> is the session timeout after each request concerning this session.</p> <p>xs:dateTime StartPoint[1][1] xs:dateTime EndPoint[0][1] tt:SearchScopeScope[1][1] tt:MetadataFilter SearchFilter[1][1] xs:int MaxMatches[0][1] xs:duration KeepAliveTime[1][1]</p>	
FindMetadataResponse	<p>Returns the <i>SearchToken</i> identifying the search session created by this request.</p> <p>tt:JobToken SearchToken[1][1]</p>	
Fault codes	Description	
env:Receiver ter:Action ter:ResourceProblem	<p>Device is unable to create a new search session.</p>	

5.14 GetMetadataSearchResults

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

GetMetadataSearchResults shall block until:

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

This operation is mandatory to support if the *MetaDataSearch* capability is set to true in the *SearchCapabilities* structure return by the *GetCapabilities* command in the Device service. If any of the specified parameters *MinResults* and *WaitTime* exceed the supported range a device shall adapt them instead of responding an error.

Table 12: GetMetadataSearchResults command

GetMetadataSearchResults	Access Class: READ_MEDIA
---------------------------------	--------------------------

Message name	Description
GetMetadataSearchResultsRequest	<p><i>SearchToken</i> specifies the search session. <i>MinResults</i> specifies the minimum number of results that should be returned. <i>MaxResults</i> specifies the maximum number of results to return. <i>WaitTime</i> defines the maximum time to block, waiting for results.</p> <p>tt:JobToken SearchToken[1][1] xs:int MinResults[0][1] xs:int MaxResults[0][1] xs:duration WaitTime[0][1]</p>
GetMetadataSearchResultsResponse	<p>Returns a structure containing the current <i>SearchState</i> and a list of <i>FindMetadataResult</i> structures.</p> <p>tt:SearchState SearchState[1][1] tt:FindMetadataResult FindMetadataResult[0][unbounded]</p>
Fault codes	Description
env:Sender ter:InvalidArgVal ter:InvalidToken	<i>The search token is invalid.</i>

5.15 GetSearchState

GetSearchState returns the current state of the specified search session.

This command is deprecated.

Table 13: GetSearchState command

GetSearchState		Access Class: READ_MEDIA
Message name	Description	
GetSearchStateRequest	<p><i>SearchToken</i> specifies the search session.</p> <p>tt:JobToken SearchToken [1][1]</p>	
GetSearchStateResponse	<p>Returns the current state of the search session.</p> <p>tt:SearchState State [1][1]</p>	
Fault codes	Description	
env:Sender ter:InvalidArgVal ter:InvalidToken	<i>The search token is invalid.</i>	

5.16 EndSearch

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

Table 14: EndSearch command

EndSearch		Access Class: READ_MEDIA
Message name	Description	
EndSearchRequest	<i>SearchToken specifies the search session.</i> tt:JobToken SearchToken [1][1]	
EndSearchResponse	<i>Returns the point in time where the search was at when ended.</i> xs:dateTime EndPoint [1][1]	
Fault codes	Description	
env:Sender ter:InvalidArgVal ter:InvalidToken	<i>The search token is invalid.</i>	

5.17 Capabilities

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

MetadataSearch Indication if the device supports generic search of recorded metadata,

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

Table 15: GetServiceCapabilities command

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

5.18 Service specific fault codes

The table below lists the Recording Search service specific fault codes. Additionally, each command can also generate a generic fault as defined in the ONVIF Core specification.

Table 16: Recording Search service specific fault codes

Fault Code	Parent Subcode	Fault Reason	Description
	Subcode		
env:Receiver	ter:Action	Resource problem encountered	The device is unable to create a new search session due to available resource limitation.
	ter:ResourceProblem		
env:Receiver	ter:ActionNotSupported	Optional Action Not Implemented	The requested action is optional and is not implemented by the device.
	ter:NotImplemented		
env:Sender	ter:InvalidArgVal	Invalid token	The specified RecordingToken / SearchToken is invalid.
	ter:InvalidToken		
env:Sender	ter:InvalidArgVal	No available recording	No recording available.
	ter:NoRecording		
env:Sender	ter:InvalidArgVal	Invalid source	The recording source is invalid.
	ter:InvalidSource		

5.19 Recording Event Descriptions

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

The time of all recording events shall specify the actual time relating to recording regardless of the sending time of the event.

```

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

```

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

```

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

```

```
</tt:Data>
</tt:MessageDescription>
```

This message signals when the data for a track is present. When the data becomes present, a message with `IsDataPresent` set to `TRUE` shall be sent. When the data becomes unavailable, The message with `IsDataPresent` set to `FALSE` shall be sent.

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

Topic: `tns1:RecordingHistory/Track/VideoParameters`

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

Topic: `tns1:RecordingHistory/Track/AudioParameters`

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

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

5.20 XPath dialect

This section defines the XPATH dialect that a device that realises the search service shall implement to parse the XPath strings that are passed to the methods of the search service.

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

```
[1] Expression ::= BoolExpr | Expression 'and' Expression
                | Expression 'or' Expression | '('Expression')' |
                'not' '('Expression')
```



```

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

```

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

```

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

```

5.21 Service specific data types

5.21.1 SearchScope

A structure for defining a limited scope when searching in recorded data.

```

<xs:complexType name="SearchScope">
  <xs:element name="IncludedSources" type="tt:SourceReference
    minOccurs="0" maxOccurs="unbounded"/>
  <xs:element name="IncludedRecordings" type="tt:RecordingReference"
    minOccurs="0" maxOccurs="unbounded"/>
  <xs:element name="RecordingInformationFilter" type="tt:XPathExpression"
    minOccurs="0"/>
</xs:complexType>

```

- **IncludedSources**
A list of sources that are included in the scope. If this list is included, only data from one of these sources shall be searched.
- **IncludedRecordings**
A list of recordings that are included in the scope. If this list is included, only data from one of these recordings shall be searched.
- **RecordingInformationFilter**
An xpath expression used to specify what recordings to search. Only those recordings with an RecordingInformation structure that matches the filter shall be searched.
- **Extension**
Extension point

5.21.2 EventFilter

```

<xs:complexType name="EventFilter">
  <xs:extension base="wsnt:FilterType"/>
</xs:complexType>

```

5.21.3 PTZPositionFilter

```

<xs:complexType name="PTZPositionFilter">

```

```
<xs:element name="MinPosition" type="tt:PTZVector"/>
<xs:element name="MaxPosition" type="tt:PTZVector"/>
<xs:element name="EnterOrExit" type="xs:boolean"/>
</xs:complexType>
```

- **MinPosition**
The lower boundary of the PTZ volume to look for.
- **MaxPosition**
The upper boundary of the PTZ volume to look for.
- **EnterOrExit**
If true, search for when entering the specified PTZ volume.

5.21.4 MetadataFilter

```
<xs:complexType name="MetadataFilter">
  <xs:element name="MetadataStreamFilter" type="tt:XPathExpression"/>
</xs:complexType>
```

- **MetadataStreamFilter**

5.21.5 FindRecordingResultList

```
<xs:complexType name="FindRecordingResultList">
  <xs:element name="SearchState" type="tt:SearchState"/>
  <xs:element name="RecordingInformation" type="tt:RecordingInformation"
    minOccurs="0" maxOccurs="unbounded"/>
</xs:complexType>
```

- **SearchState**
The state of the search when the result is returned. Indicates if there can be more results, or if the search is completed.
- **RecordingInformation**
A RecordingInformation structure for each found recording matching the search.

5.21.6 FindEventResultList

```
<xs:complexType name="FindEventResultList">
  <xs:element name="SearchState" type="tt:SearchState"/>
  <xs:element name="Result" type="tt:FindEventResult" minOccurs="0"
    maxOccurs="unbounded"/>
</xs:complexType>
```

- **SearchState**
The state of the search when the result is returned. Indicates if there can be more results, or if the search is completed.
- **Result**
A FindEventResult structure for each found event matching the search.

5.21.7 FindEventResult

```
<xs:complexType name="FindEventResult">
  <xs:element name="RecordingToken" type="tt:RecordingReference"/>
  <xs:element name="TrackToken" type="tt:TrackReference"/>
  <xs:element name="Time" type="xs:dateTime"/>
  <xs:element name="Event" type="wsnt:NotificationMessageHolderType"/>
  <xs:element name="StartStateEvent" type="xs:boolean"/>
</xs:complexType>
```

- **RecordingToken**
The recording where this event was found. Empty string if no recording is associated with this event.

- **TrackToken**
A reference to the track where this event was found. Empty string if no track is associated with this event.
- **Time**
The time when the event occurred.
- **Event**
The description of the event.
- **StartStateEvent**
If true, indicates that the event is a virtual event generated for this particular search session to give the state of a property at the start time of the search.

5.21.8 FindPTZPositionResultList

```
<xs:complexType name="FindPTZPositionResultList">
  <xs:element name="SearchState" type="tt:SearchState"/>
  <xs:element name="Result" type="tt:FindPTZPositionResult minOccurs="0"
    maxOccurs="unbounded"/>
</xs:complexType>
```

- **SearchState**
The state of the search when the result is returned. Indicates if there can be more results, or if the search is completed.
- **Result**
A FindPTZPositionResult structure for each found PTZ position matching the search.

5.21.9 FindPTZPositionResult

```
<xs:complexType name="FindPTZPositionResult">
  <xs:element name="RecordingToken" type="tt:RecordingReference"/>
  <xs:element name="TrackToken" type="tt:TrackReference"/>
  <xs:element name="Time" type="xs:dateTime"/>
  <xs:element name="Position" type="tt:PTZVector"/>
</xs:complexType>
```

- **RecordingToken**
A reference to the recording containing the PTZ position.
- **TrackToken**
A reference to the metadata track containing the PTZ position.
- **Time**
The time when the PTZ position was valid.
- **Position**
The PTZ position.

5.21.10 FindMetadataResultList

```
<xs:complexType name="FindMetadataResultList">
  <xs:element name="SearchState" type="tt:SearchState"/>
  <xs:element name="Result" type="tt:FindMetadataResult minOccurs="0"
    maxOccurs="unbounded"/>
</xs:complexType>
```

- **SearchState**
The state of the search when the result is returned. Indicates if there can be more results, or if the search is completed.
- **Result**
A FindMetadataResult structure for each found set of Metadata matching the search.

5.21.11 FindMetadataResult

```
<xs:complexType name="FindMetadataResult">  
  <xs:element name="RecordingToken" type="tt:RecordingReference"/>  
  <xs:element name="TrackToken" type="tt:TrackReference"/>  
  <xs:element name="Time" type="xs:dateTime"/>  
</xs:complexType>
```

- **RecordingToken**
A reference to the recording containing the metadata.
- **TrackToken**
A reference to the metadata track containing the matching metadata.
- **Time**
The point in time when the matching metadata occurs in the metadata track.

Annex A. Revision History

Rev.	Date	Editor	Changes
2.1	Jul-2011	Hans Busch	Split from Core 2.0
2.1.1	Jan-2012	Hans Busch	Change Requests 325, 327, 328, 435, 535
2.2	Apr-2012	Hans Busch	Change Requests 601, 608, 614, 616, 621, 671, 672
2.2.1	Sep-2012	Hans Busch	Change Requests 708, 765, 766, 778, 780, 781, 827, 837
2.3	May-2013	Michio Hirai	Change Request 884