# ONVIF™
# Receiver Service Specification

Version 18.12
December, 2018

# CONTENTS

# 1 Scope

This document defines the web service interface for configuration a network streaming receiver. The receiver is used for displaying and recording audio video streams. Additionally the associated events are defined.

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

# 2 Normative references

ONVIF Core Specification

<http://www.onvif.org/specs/core/ONVIF-Core-Specification.pdf>

ONVIF Streaming Specification

<http://www.onvif.org/specs/stream/ONVIF-Streaming-Spec.pdf>

# 3 Terms and Definitions

## 3.1 Definitions

| | |
|---|---|
| **Metadata** | All streaming data except video and audio, including video analytics results, PTZ position data and other metadata (such as textual data from POS applications). |

## 3.2 Abbreviations

| | |
|---|---|
| JPEG | Joint Photographic Expert Group |
| MPEG-4 | Moving Picture Experts Group - 4 |
| RTSP | Real Time Streaming Protocol |

# 4 Overview

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

A receiver can operate in three distinct modes:

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

NeverConnect. The receiver does not attempt to connect.

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

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

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

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

# 5 Service

This service offers commands to manage Receiver objects, which are used to receive media streams from other devices. A Receiver object contains the information how to setup the stream, the mode of the receiver and the Stream Uri (MediaUri). A device shall at least support Media Uris of 128 octet length. The Receiver - MaximumRTSPURILength capability indicates the maximum length supported by the device. The Receiver Service shall be implemented by devices that can receive media streams.

The IP or DNS address in the transmit URI given to the receiver, is the address that the device hosting the receiver service will use to access the transmit device. If, for example, the client has to communicate through a NAT router to access the transmitter and the receiver, the transmitter address that the client gives the receiver (in this case a local network address) may not be the same address that the client would use to access the transmitter (in this case an external network address).

A device shall support RTP transfer via UDP and RTP transfer via RTSP/HTTP/TCP, see ONVIF Streaming Specification. A device may support other RTP transport protocols and shall indicate what it supports with the appropriate capability, see 5.5.

## 5.1 Persistence

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

## 5.2 Receiver modes

A receiver can operate in three distinct modes:

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

NeverConnect. The receiver does not attempt to connect.

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

### 5.3 Synchronization Points

Because receivers use RTSP addresses to specify the source of the stream, they do not necessarily have access to the web services interface of the transmitter. This means that they cannot use the SetSynchronizationPoint command described in the section ”Synchonization Point” of the ONVIF Streaming Specification..

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

## 5.4 Receiver commands

This section describes the commands offered by the Receiver Service.

### 5.4.1 GetReceivers

This operation lists all receivers that currently exist on the device. The Receiver Service shall support this command.

REQUEST:

This is an empty message.

RESPONSE:

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

FAULTS:

None

ACCESS CLASS:

- **READ MEDIA**

### 5.4.2 GetReceiver

This operation retrieves the details of a specific receiver whose token is known to the client. The Receiver Service shall support this command.

REQUEST:

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

RESPONSE:

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

FAULTS:

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

ACCESS CLASS:

- **READ MEDIA**

### 5.4.3 CreateReceiver

This operation creates a new receiver. The Receiver Service shall support this command.

REQUEST:

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

RESPONSE:

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

FAULTS:

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

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

ACCESS CLASS:

- **ACTUATE**

### 5.4.4 DeleteReceiver

This operation deletes an existing receiver. A device may reject deletion of a receiver which is in use. The Receiver Service shall support this command.

REQUEST:

- **ReceiverToken [tt:ReferenceToken]**
  The token of the receiver to be deleated.

RESPONSE:

This is an empty message.

FAULTS:

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

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

ACCESS CLASS:

- **ACTUATE**

### 5.4.5 ConfigureReceiver

This operation configures a receiver. The Receiver Service shall support this command.

REQUEST:

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

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

RESPONSE:

This is an empty message.

FAULTS:

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

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

ACCESS CLASS:

- **ACTUATE**

### 5.4.6 SetReceiverMode

This operation may be used to set the mode of the receiver independently of the rest of its configuration. The Receiver Service shall support this command.

REQUEST:

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

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

RESPONSE:

This is an empty message.

FAULTS:

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

ACCESS CLASS:

- **ACTUATE**

### 5.4.7 GetReceiverState

This operation determines whether the receiver is currently disconnected, connected or attempting to connect. The Receiver Service shall support this command.

REQUEST:

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

RESPONSE:

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

FAULTS:

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

ACCESS CLASS:

- **READ MEDIA**

### 5.5 GetServiceCapabilitites

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

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

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

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

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

**MaximumRTSPURILength:** The maximum length allowed for RTSP URIs.

REQUEST:

This is an empty message.

RESPONSE:

- **Capabilities [trv:Capabilities]**
  List of capabilties as defined above.

FAULTS:

None

ACCESS CLASS:

- **PRE_AUTH**

## 5.6 Events

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

### 5.6.1 ChangeState

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

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

### 5.6.2 Connection Failed

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

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

## Annex A. Revision History

| Rev. | Date | Editor | Changes |
|---|---|---|---|
| 2.1 | Jul-2011 | Hans Busch | Split from Core 2.0<br>Change Request 173 |
| 2.1.1 | Jan-2012 | Hans Busch | Change Request 535 |
| 2.2 | May-2012 | Hans Busch | Change Request 677 |
| 2.2.1 | Dec-2012 | Hans Busch | Change Request 708, 709 |
| 17.06 | Jun 2017 | Stefan Andersson<br>Hiroyuki Sano | Update method layouts, Change Request 1843, 2089<br>Change Request 2125, 2126 |
| 18.12 | Dec-2018 | Hiroyuki Sano | Change Request 2378 |