ONVIF™
WebRTC Specification

Version 25.12

December, 2025

CONTENTS

1 Scope **4**

2 Normative references **4**

3 Terms and Definitions **5**

3.1 Definitions 5

3.2 Abbreviations 5

4 Overview **5**

5 Signaling Protocol **6**

5.1 WebSocket Connection Management 7

5.2 Communication Protocol 7
5.2.1 Session Capabilities 8
5.2.2 register 8
5.2.2.1 register - Client to Signaling Server 8
5.2.2.2 register - Device to Signaling Server 8
5.2.3 unregister 9
5.2.4 connect 9
5.2.4.1 connect - Client to Signaling Server 9
5.2.4.2 connect - Signaling Server to Device 10
5.2.5 invite 10
5.2.6 trickle 11
5.2.7 extend 12
5.2.7.1 extend - Client to Signaling Server 12
5.2.7.2 extend - Device to Signaling Server 12
5.2.8 error 12
5.2.9 Example Flow (informative) 13

6 WebRTC Usage **14**

6.1 Bandwidth management 14

6.2 Feedback mechanisms 14

6.3 Congestion control 14

6.4 ICE candidates 14

6.5 Video 15

6.6 Audio 15

6.7 Data channels 15
6.7.1 Enabling data channels 15
6.7.2 Metadata over data channels 15

Annex A Bibliography **16**

Annex B Revision History **17**

1 Scope

This document defines how WebRTC and related protocols are to be used for ONVIF clients and devices to establish a peer-to-peer connection between a client and a device using a signaling server.

The client communication with its signaling server is provided as an example, both those components belong to the Client and their interaction may be vendor dependent.

Sending PTZ and other commands via WebRTC datachannels is outside of the scope of this version of the specification.

2 Normative references

IETF RFC 4585 - Extended RTP Profile for Real-Time Transport Control Protocol (RTCP)-Based Feedback
<https://datatracker.ietf.org/doc/html/rfc4585>

IETF RFC 5104 - Codec Control Messages in the RTP Audio-Visual Profile with Feedback (AVPF) Profile
<https://datatracker.ietf.org/doc/html/rfc5104>

IETF RFC 5245 - Interactive Connectivity Establishment (ICE)
<http://tools.ietf.org/html/rfc5245>

IETF RFC 6544 - TCP Candidates with Interactive Connectivity Establishment (ICE)
<http://tools.ietf.org/html/rfc6544>

IETF RFC 6455 - The WebSocket Protocol
<http://tools.ietf.org/html/rfc6455>

IETF RFC 6749 - The OAuth 2.0 Authorization Framework
<http://tools.ietf.org/html/rfc6749>

IETF RFC 6750 - The OAuth 2.0 Authorization Framework: Bearer Token Usage
<http://tools.ietf.org/html/rfc6750>

IETF RFC 7064 - URI Scheme for the Session Traversal Utilities for NAT (STUN) Protocol
<https://datatracker.ietf.org/doc/html/rfc7064>

IETF RFC 7065 - Traversal Using Relays around NAT (TURN) Uniform Resource Identifiers
<https://datatracker.ietf.org/doc/html/rfc7065>

IETF RFC 8829 - JavaScript Session Establishment Protocol (JSEP)
<http://tools.ietf.org/html/rfc8829>

IETF RFC 8834 - Media Transport and Use of RTP in WebRTC
<http://tools.ietf.org/html/rfc8834>

IETF RFC 8839 - Session Description Protocol (SDP) Offer/Answer Procedures for Interactive Connectivity Establishment (ICE)
<http://tools.ietf.org/html/rfc8839>

IETF RFC 8840 - A Session Initiation Protocol (SIP) Usage for Incremental Provisioning of Candidates for the Interactive Connectivity Establishment (Trickle ICE)
<http://tools.ietf.org/html/rfc8840>

IETF RFC 8445 - Interactive Connectivity Establishment (ICE)
<http://tools.ietf.org/html/rfc8445>

IETF RFC 8489 - Session Traversal Utilities for NAT (STUN)
<http://tools.ietf.org/html/rfc8489>

IETF RFC 8656 - Traversal Using Relays around NAT (TURN)
<http://tools.ietf.org/html/rfc8656>

JSON-RPC 2.0
<https://www.jsonrpc.org/specification>

ONVIF Security Service Specification
<https://www.onvif.org/specs/srv/security/ONVIF-Security-Service-Spec.pdf>

OpenID Connect Core
<https://openid.net/specs/openid-connect-core-1_0.html>

W3C WebRTC Specification: Real-Time Communication in Browsers
<https://www.w3.org/TR/webrtc/>

3 Terms and Definitions

3.1 Definitions

Signaling Server	A server that manages the WebRTC connections between clients and devices.
Session ID	Identifier for a connection between peers.

3.2 Abbreviations

ICE	Interactive Connectivity Establishment
NAT	Network Address Translation
SDP	Session Description Protocol
STUN	Session Traversal Utilities for NAT
TURN	Traversal Using Relays around NAT
WebRTC	Web Real-Time Communication

4 Overview

The WebRTC standard includes APIs for communicating with an ICE Agent, but the signaling component is not part of it. Signaling is needed in order for two peers to share how they should connect.

Signaling can be implemented in many different ways, and the WebRTC standard doesn't recommend any specific solution.

This specification contains documentation and examples of the signaling protocol used in ONVIF to set up a WebRTC peer-to-peer connection. The setup involves three participants: *client*, *device* and *signaling server*.

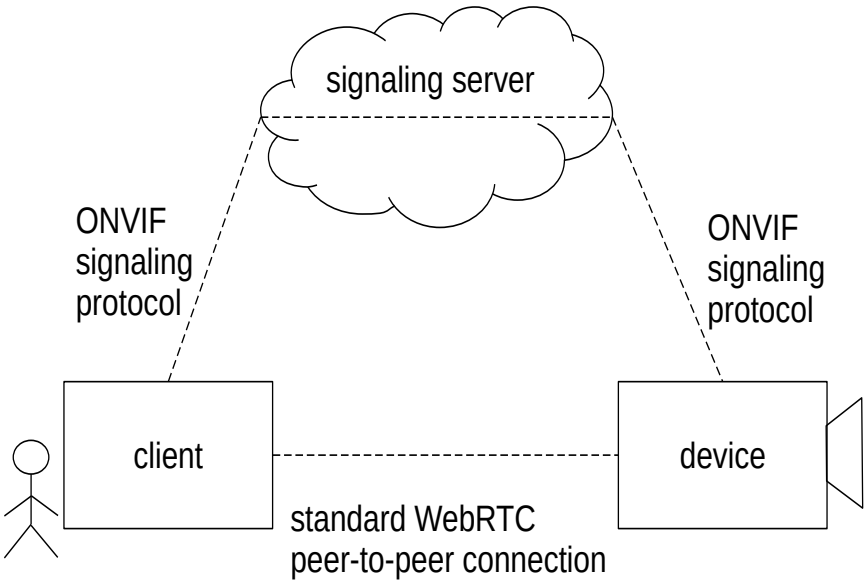

Figure 1: Client, device, signaling server

In Figure 1

- The *client* represents a user who initiates the WebRTC session.
- The *device* is the resource delivering the media, for example a camera.
- The *signaling server* is the mediator that both *client* and *device* connect to in order to establish a peer-to-peer connection between them.

The *Signaling Protocol* described in this specification details the data exchange between client and device via the signaling server.

Once a peer-to-peer connection has been established, how WebRTC is used is described in the WebRTC usage chapter.

5 Signaling Protocol

The *Signaling Protocol* defines the messages between a *client* and a *device* with the intention of establishing a WebRTC peer-to-peer connection between the client and a device. The messages are always sent via the *signaling server*, called *server* from now on. Once a peer-to-peer connection has been established the connection with the server can be dropped without affecting the peer-to-peer connection.

A WebRTC signaling server may enforce expiration on active sessions by using the ExpiryTime property of the connect command. In such scenarios, the client shall maintain the connection with the signaling *server* and send extend command periodically before the session expires.

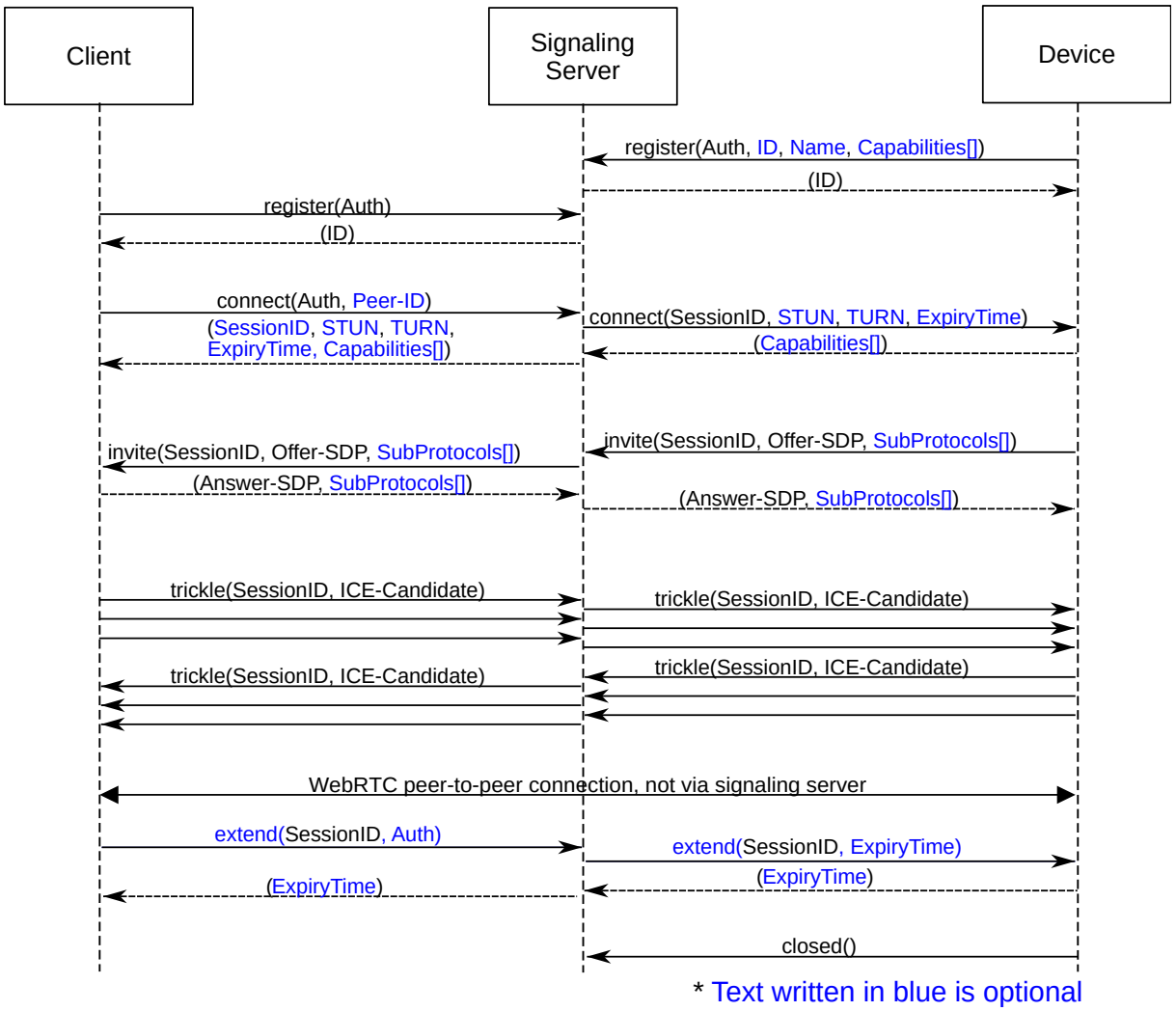

Figure 2: Signaling flow sequence diagram

Figure 2 shows an overview of the messages being sent between client, signaling server and device.

5.1 WebSocket Connection Management

Both a client and a device need to connect to the server by setting up a WebSocket connection according to RFC 6455. The WebSocket sub protocol shall be set to 'webrtc.onvif.org'.

A device shall connect to a signaling server as soon as the configuration contains a valid URI.

In case a connection is dropped the device shall reconnect automatically. Each client should use an individual ascending interval strategy to avoid that all clients reconnect at the same time.

A client as well as a device shall provide their access token using the Authorization header or via query parameter on the WebSocket URI as defined by RFC 6750 Section 2.3.

Once the WebSocket is open, both device and client shall immediately send a register command as specified in section 5.2.2 when they receive the HTTP 101 switching protocol.

5.2 Communication Protocol

The communication protocols shall use JSON-RPC version 2 over a WebSocket connection . Parameters shall be passed through an Object by-name as defined in JSON-RPC section 4.2. Table 1 provides the encodings to be used.

Table 1: Parameter encoding

Type	Encoding
SDP	SDP according to RFC 8866 with \r\n escaping
IceCandidate	JSON encoding according to W3C WebRTC RTCIceCandidateInit
RTCIceServer	Connection parameters for a STUN or TURN server as defined by W3C WebRTC
DataChannelSubprotocol	Data channel subprotocol identifier

5.2.1 Session Capabilities

When calling connect command, an optional parameter called capabilities may be returned. The capabilities parameter will inform the client of the functionality available in the session. Table 2 list the possible capabilities

Table 2: Capabilities

Capability	Functionality
subProtocols	SubProtocols are supported, read section 6.7. As Sub-protocols are added before capabilities, some implementations may support this without it being part of the capabilities. Support of these capabilities can be seen within the invite command received.
extend	The Connect command will return an expiry time parameter. It is possible to call the extend command before the expiry time ends to extend the session.

5.2.2 register

A signaling server shall expect this command to be sent once before any other command is sent over the WebSocket connection. The signaling server shall verify the validity of the access token either provided by the http connect request or as request parameter. Token details are outside of the scope of this specification.

5.2.2.1 register - Client to Signaling Server

When the client sends the `register` command to the signaling server

REQUEST:

- **authorization optional [string]**
  Access token that authorizes the device or client.

RESPONSE:

- **id [string]**
  The ID assigned by the signaling server to the endpoint.

FAULTS:

- **401 Authorization failed**
  The access token cannot be verified or has expired.

5.2.2.2 register - Device to Signaling Server

When the device sends the `register` command to the signaling server

REQUEST:

- **authorization optional [string]**
  Access token that authorizes the device or client.

- **name optional [string]**
  The human readable name of the device or client.
- **capabilities optional [string[]]**
  Capabilities of the device. This should only be used by the device and the possible values can be found in the section 5.2.1.

RESPONSE:

- **id [string]**
  The ID assigned by the signaling server to the endpoint.

FAULTS:

- **401 Authorization failed**
  The access token cannot be verified or has expired.

5.2.3 unregister

A client or device may send this to a signaling server before closing the WebSocket connection, indicating that the closing of the WebSocket is intentional and that it no longer wants to be registered. The main use case is to support a device or client that isn't always connected to the network.

REQUEST:

<none>

RESPONSE:

<none>

FAULTS:

- **404 Not Found**
  Not registered.

5.2.4 connect

An ONVIF compliant signaling server and device shall support this command to establish a streaming session between two peers.

5.2.4.1 connect - Client to Signaling Server

When the client sends the `connect` command to the signaling server

REQUEST:

- **peer [string]**
  The ID of the peer the client wants to connect to. This ID is defined outside of this specification and must be negotiated out-of-band.
- **authorization [string]**
  Access token of the client to be authorized by the signaling server.
- **profile optional [string]**
  Token of the media streaming profile to use.

RESPONSE:

- **session [string]**
  The ID assigned by the signaling server to the session.
- **iceServers optional unbounded [RTCIceServer]**
  List of STUN and TURN servers provided by the signaling server to be used for this session.

- **expiryTimeSeconds optional [integer]**
  The time represented in seconds before the session ends.
- **capabilities optional [string[]]**
  A list of capabilities supported for a session.

FAULTS:

- **401 Authorization failed**
  The access token cannot be verified or has expired.
- **403 Forbidden**
  The client is not authorized to connect to the provided peer.
- **480 Temporary unavailable.**
  The target device is currently unavailable.

5.2.4.2 connect - Signaling Server to Device

When the signaling server forwards the `connect` command to the device

REQUEST:

- **session [string]**
  The ID assigned by the signaling server to the session.
- **profile optional [string]**
  Token of the media streaming profile to use.
- **iceServers optional unbounded [RTCIceServer]**
  List of STUN and TURN servers provided by the signaling server to be used for this session.
- **expiryTimeSeconds optional [integer]**
  The time represented in seconds before the session ends.

RESPONSE:

- **capabilities optional [string[]]**
  A list of capabilities supported for a session. This list shall be the same as the list provided with the `register` command.

FAULTS:

- **480 Temporary unavailable.**
  The target device is currently unavailable.

An ONVIF compliant signaling server shall provide the session ID and a list of STUN and TURN servers in both request and response such that both device and client can use them. A compliant client and device shall support password based authentication of TURN server. The urls parameter of the iceServers shall always be encoded as array of strings. The optional shortcut defined by W3C WebRTC specification of passing a single string shall not be used.

An ONVIF compliant device shall issue an invite method immediately after responding to this method.

5.2.5 invite

An ONVIF compliant signaling server shall support this command to establish a streaming session between two peers. It shall forward this command to the client and correspondingly route the response back to the device.

The dynamics and the format of the SDP records are defined in RFC 8829.

Usage of WebRTC data channels is specified in section 6.7.

REQUEST:

- **session [string]**
  The ID assigned by the signaling server to the session.
- **offer [SDP]**
  The SDP offer provided by the device.
- **subprotocols optional unbounded [DataChannelSubprotocol]**
  List of data channel subprotocols to be allowed

RESPONSE:

- **answer [SDP]**
  The SDP answer provided by the client.
- **subprotocols optional unbounded [DataChannelSubprotocol]**
  List of data channel subprotocols to be allowed

FAULTS:

- **400 Bad Request**
  Invalid session ID or SDP payload.
- **408 Request Timeout**
  The peer did not react.
- **410 Gone**
  The peer is no more available.

5.2.6 trickle

An ONVIF compliant signaling server, device and client shall support this command to signal new ICE candidates for the trickleICE procedure as defined in RFC 8840.

A signaling server shall relay this command unaltered to the peer.

REQUEST:

- **session [string]**
  The ID assigned by the signaling server to the session.
- **candidate [IceCandidate]**
  The ICE Candidate SDP update for the peer.

RESPONSE:

<none>

FAULTS:

<none>

Both client and device produce ICE candidates and send them to each other, via the server. Once all ICE candidates have been sent by a peer, it shall send a last trickle notification with an empty ICE candidate content to indicate that all candidates have been sent according to RFC 8838.

NOTE: Note that ICE candidates can arrive **before** the SDP offer and the implementing client needs to handle this.

If everything works as it should, a peer-to-peer WebRTC session can be set up between client and device. After the session has been established the client can terminate the WebSocket session to the signaling server and the peer-to-peer connection will not be affected. The only reason to keep the connection to the server is if the client needs to extend the session.

5.2.7 extend

This command updates a session if more time is required. If an expiryTime is given during the connect command response, the session will timeout. It is possible to request an extension by just sending a new authorization token to the signaling server.

5.2.7.1 extend - Client to Signaling Server

When the client sends the `extend` command to the signaling server

REQUEST:

- **session [string]**
  The ID assigned by the signaling server to the session.
- **authorization optional [string]**
  Access token that authorizes the client.

RESPONSE:

- **expiryTimeSeconds [integer]**
  The time represented in seconds until the session expires.

FAULTS:

- **401 Authorization failed**
  The authorization token cannot be verified. The authorization token may not include the required claims, or has expired.
- **403 Forbidden**
  The client is not authorized to connect to the provided peer.

5.2.7.2 extend - Device to Signaling Server

When the device sends the `extend` command to the signaling server

REQUEST:

- **session [string]**
  The ID assigned by the signaling server to the session.
- **expiryTimeSeconds optional [integer]**
  The new expiry time requested by the signaling server represented in seconds.

RESPONSE:

- **expiryTimeSeconds [integer]**
  The time represented in seconds until the session expires.

FAULTS:

- **401 Authorization failed**
  The authorization token cannot be verified. The authorization token may not include the required claims, or has expired.
- **403 Forbidden**
  The client is not authorized to connect to the provided peer.

5.2.8 error

An ONVIF compliant signaling server, device and client shall support sending or receiving notifications signaling that an error has occurred.

This message is a "Response" or "Notification" message as defined by JSON-RPC and shall contain "Error object" as defined by JSON-RPC. If the error is not connected to a request, but sent as a notification, the id shall be omittedt.

REQUEST:

- **code [int]**
  A number that indicates the error type that occurred.
- **message [string]**
  A string providing a short description of the error. The message should be limited to a concise single sentence.
- **session [string]**
  The ID assigned by the signaling server to the session.

RESPONSE:

<none>

FAULTS:

<none>

Table 3 defines possible error codes:

Table 3: WebRTC signaling errors

Signaling Error	Error Code	Reason
Insufficient resources	1001	The peer does not have sufficient resources.
Peer disconnected	1002	A peer in the session has closed its websocket connection with the Signaling Server.
Malformed candidate	1003	The trickle ICE candidate received is malformed.

Note that the message layout slightly deviates from the JSON RPC 2.0 specification.

5.2.9 Example Flow (informative)

The following example shows the flow using JSON-RPC 2.0. Note that for improved readability the mandatory JSON version string is not shown.

```
Client -> Server: { "method": "register", "params": {"authorization": "access token"}, "id":1}
Server -> Client: { "result": { "id": "client-a1" }, "id": 1}
Device -> Server: { "method": "register", "params": {"authorization": "access token"}, "id":1}
Server -> Device: { "result": { "id": "device-b1" }, "id": 1}

Client -> Server: { "method": "connect",
                "params": {"authorization": "access token", "peer": "device-b1"}, "id":2}
Server -> Device: { "method": "connect",
                "params": {"session": "s1", "iceServers": [{"urls": ["stun:1.2.3.4"]}],
                "id":2}
Device -> Server: { "result": {}, "id": 2}
Server -> Client: { "result": {"session": "s1",
                "iceServers": [{"urls": ["stun:1.2.3.4"]}]}, "id": 2}
```

```
Device -> Server: { "method": "invite", "params": {"session": "s1", "offer": "SDP...",
                "subprotocols": ["proto1"]}, "id":3}
Server -> Client: { "method": "invite", "params": {"session": "s1", "offer": "SDP...",
                "subprotocols": ["proto1"]}, "id":3}
Client -> Server: { "result": {"answer": "SDP...", "subprotocols": ["proto1"]}, "id": 3}
Server -> Device: { "result": {"answer": "SDP...", "subprotocols": ["proto1"]}, "id": 3}

Device -> Server: { "method": "trickle", "params": {"session": "s1", "candidate": {...}}}
Server -> Client: { "method": "trickle", "params": {"session": "s1", "candidate": {...}}}

Client -> Server: { "method": "trickle", "params": {"session": "s1", "candidate": {...}}}
Server -> Device: { "method": "trickle", "params": {"session": "s1", "candidate": {...}}}

Client -> Server: { "method": "trickle", "params": {"session": "s1", "candidate": {...}}}
Server -> Device: { "method": "trickle", "params": {"session": "s1", "candidate": {...}}}
...
Client -> Server: { "method": "extend", "params": {"session": "s1", "authorization": "...",
                "id": 4}
Server -> Client: { "error": {"code": -32601, "message": "Method not found"}, "id": "4"}
...
Device -> Server: { "error": {"code": 1001, "message": "Insufficient resources"}}
Server -> Client: { "error": {"code": 1001, "message": "Insufficient resources"}}
```

6 WebRTC Usage

This chapter describes ONVIF specific usage of WebRTC regarding how to send data between the client and device on the peer-to-peer connection.

6.1 Bandwidth management

Devices may define profiles for different streaming scenarios that can be chosen by the client based on the available bandwidth or other limiting factors. For example, a specific profile for mobile clients, another one for high quality streaming, etc.

Devices should estimate available uplink bandwidth and reduce stream bitrate as needed to avoid packet loss. It may be achieved by selecting video stream from another profile with the same video source and the same video codec. Once there is enough available bandwidth, devices should aim to switch back to the desired configuration.

6.2 Feedback mechanisms

Devices shall be able to exchange RTCP feedback with their peer as defined in RFC 8834. In particular, devices shall handle FIR messages, as defined in RFC 5104, and PLI messages, as defined in RFC 4585, and attempt to provide a synchronization point according to the semantics of these requests.

6.3 Congestion control

ONVIF recommends to always use congestion control to ensure that WebRTC cannot be used to flood the network. For additional information, IETF recommends to use mechanism specified in RFC8836 while many actual implementations refer to [RTP Extension for congestion control] and its out of the scope for this specification to mandate specific congestion control mechanism.

6.4 ICE candidates

A device shall support generating and receiving host, server reflexive and relay ICE candidates as defined in RFC 8445.

A device shall handle passive ICE TCP candidates as defined in RFC 6544. On reception, it shall generate an active ICE TCP candidate to match and establish a TCP connection to the endpoint announced by the candidate. A device is not required to listen on a TCP endpoint and generate a passive ICE TCP candidate.

6.5 Video

When using WebRTC with a web browser only certain codecs will work. Because of this ONVIF recommends to use the h.264 codec.

6.6 Audio

When using WebRTC with a web browser only certain codecs will work. Because of this ONVIF recommends to use the Opus or PCMU codec.

If the selected media profile contains an AudioDecoder, the device shall include an audio track with `a=recvonly` in its SDP offer. If the selected media profile contains an AudioEncoder, the device shall include an audio track with `a=sendonly in its SDP offer`. If the profile contains both AudioEncoder and AudioDecoder, the device shall include an audio track with `a=sendrecv` in its SDP offer. For two-way audio it is recommended to use echo cancellation to avoid feedback loops.

6.7 Data channels

6.7.1 Enabling data channels

If the device supports WebRTC data channels, it shall signal this in the SDP offer of the invite method. If the client supports data channels and intends to use them during the WebRTC session, it shall signal this in the SDP answer.

ONVIF-defined data channel subprotocol names shall have “vnd.onvif” prefix. Usage of this prefix is restricted to ONVIF-defined subprotocols only.

Before creating and using any data channel subprotocol, the device and client shall exchange a list of allowed data channel subprotocols in the invite method. Only subprotocols provided by both the device and client during the invite method shall be allowed to be enabled during the WebRTC session.

At any time after obtaining a WebRTC connection, a client may request enabling a pre-negotiated data channel subprotocol by creating a new data channel with the specified subprotocol identifier. Device shall enable the requested data channel subprotocol only after ensuring it was negotiated during the SDP exchange; otherwise, it shall close the data channel.

6.7.2 Metadata over data channels

Subprotocol identifier: vnd.onvif.metadata+gzip

The Metadata data channel subprotocol enables streaming XML Metadata over WebRTC data channels. An ONVIF compliant device supporting streaming of metadata shall support gzip compressed metadata as signaled via the subprotocol vnd.onvif.metadata+gzip.

A device shall use the MetadataConfiguration from the media streaming profile selected for the WebRTC session and ignore the CompressionType setting.

The data channel Metadata payload is a binary data channel message starting with a GZIP header according to RFC 1952, followed by the compressed data. Messages may be fragmented as needed, and the client shall determine the beginning of a new GZIP payload based on the existence of the GZIP header.

Annex A.
Bibliography

[RFC8836] Congestion Control Requirements for Interactive Real-Time Media
[https://datatracker.ietf.org/doc/html/rfc8836

[RTP Extension for congestion control] RTP Extensions for Transport-wide Congestion Control
[https://datatracker.ietf.org/doc/html/draft-holmer-rmcat-transport-wide-cc-extensions-01

Annex B.
Revision History

Rev.	Date	Editor	Changes
24.06	Jun 2024	Fredrik Svensson, Jonas Cremon, Karin Hedlund, Hans Busch	First release.
24.12	Dec 2024	Jean-Francois Levesque, Jose Melancon	Add requirements for device supporting ICE candidates. Add RTCP & I-Frame request requirements
25.06	June, 2025	Tomasz Zajac	Added data channels.
25.12	Dec, 2025	Henrik Hein	Added extend method and capabilities.