ONVIF™
Uplink Specification

Version 24.06

June 2024

CONTENTS

1 Scope **4**

2 Normative references **4**

3 Terms and Definitions **4**

3.1 Definitions 4

3.2 Abbreviations 4

4 Overview **4**

5 Uplink **5**

5.1 Connection Establishment 5
5.1.1 Establishing the reverse tunnel 5
5.1.1.1 Native h2c-reverse 5
5.1.1.2 WebSocket h2c-reverse 6
5.1.1.3 HTTP/2 Frames 6
5.1.1.4 HTTP Transactions 7

5.2 Connection Management 7

5.3 Authentication 7
5.3.1 mTLS authentication 7
5.3.2 JWT authentication 7

5.4 Media Streaming 8
5.4.1 Introduction 8
5.4.2 RTSP over HTTP 8
5.4.3 RTSP over WebSockets 8

6 Configuration Interface **8**

6.1 Configuration parameters 8

6.2 GetUplinks 8

6.3 SetUplink 9

6.4 DeleteUplink 9

6.5 Capabilities 10

Annex A Revision History **11**

1 Scope

This document defines the connection protocol for connecting a web service behind a firewall to a client reachable in the internet.

2 Normative references

IETF RFC 5246 - The Transport Layer Security (TLS) Protocol, Version 1.2
<http://tools.ietf.org/html/rfc5246>

IETF RFC 6125 - Representation and Verification of Domain-Based Application Service

Identity within Internet Public Key Infrastructure Using X.509 (PKIX)

Certificates in the Context of Transport Layer Security (TLS)
<https://tools.ietf.org/html/rfc6125>

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

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

IETF RFC 7540 - Hypertext Transfer Protocol Version 2 (HTTP/2)
<https://tools.ietf.org/html/rfc7540>

IETF RFC 8441 - Bootstrapping WebSockets with HTTP/2
<https://tools.ietf.org/html/rfc8441>

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

ONVIF Media2 Service Specification
<https://www.onvif.org/specs/srv/media/ONVIF-Media2-Service-Spec.pdf>

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

Apple Computer Inc. RTSP over HTTP, Tunneling QuickTime RTSP and RTP over HTTP
<https://opensource.apple.com/source/QuickTimeStreamingServer/QuickTimeStreamingServer-412.42/Documentation/RTSP_Over_HTTP.pdf>

3 Terms and Definitions

3.1 Definitions

Local Service	A service to be used by a client behind a firewall.
Remote Client	A client that wants to access a service that is located behind a firewall.
Uplink	The connection establish by the local service to the remote client.

3.2 Abbreviations

HTTP	Hypertext Transfer Protocol
TLS	Transport Layer Security

4 Overview

The ONVIF connection protocols base on the standard web service model where the client initiates a connection to a device as depicted in Figure 1..

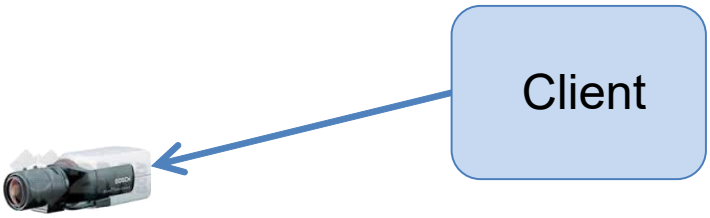

Figure 1: Standard connection initiated from the client

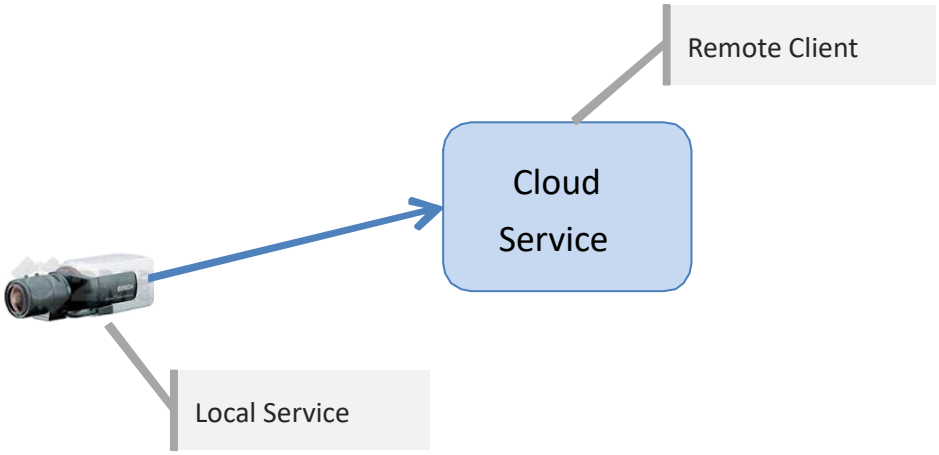

Figure 2: Connection initiation from the device

This document specifies a solution that allows a camera to use an uplink to facilitate existing web server and RTSP server functionality using the http/2 protocol.

5 Uplink

5.1 Connection Establishment

There are two connection mechanisms available for a device to connect to a remote client. The device can either use an http/2 reverse tunnel or a WebSocket.

5.1.1 Establishing the reverse tunnel

The device initiates the connection to the cloud service. This is done in three phases. Phase 2 and 3 depend on the protocol being used and are detailed below. In the first phase the device acts as TLS client that connects to the cloud service. Details of the TCP and TLS exchange are out of scope of this specification.

5.1.1.1 Native h2c-reverse

The second phase includes the connection upgrade to HTTP/2 h2c-reverse protocol which is confirmed by the cloud service with a 101 HTTP response. The third phase of the connection then fully complies to an HTTP2 connection as if it were initiated from the cloud service. Please note that the figure only shows the most relevant packets.

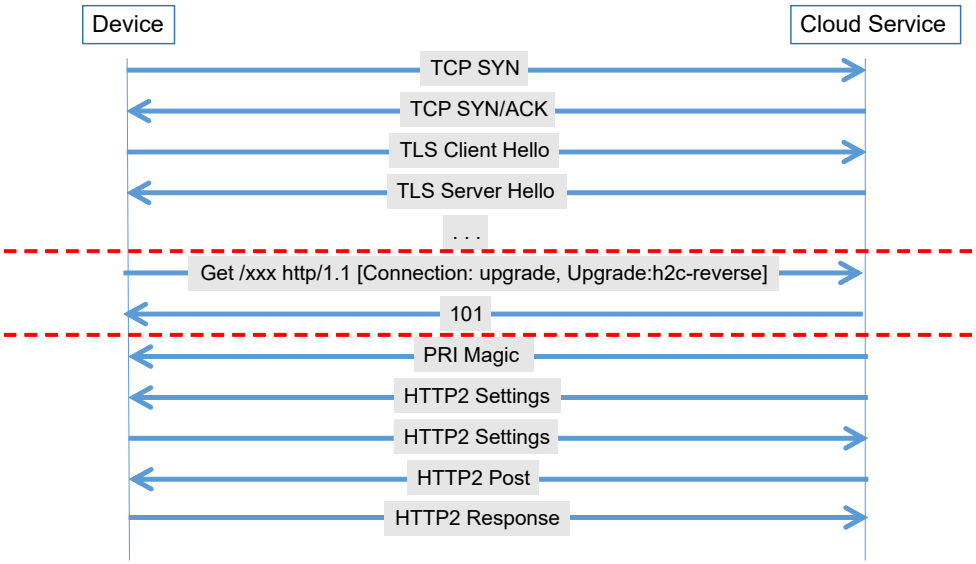

Figure 3: Connection initiation from the device

This protocol shall be used when the uri protocol states 'https'. A device signaling support for https via its Protocol capability shall support native h2c-reverse.

5.1.1.2 WebSocket h2c-reverse

The second phase includes the connection upgrade to a WebSocket with the reverse HTTP2 subprotocol which is confirmed by the cloud service with a 101 HTTP response. The third phase of the connection then corresponds to an HTTP2 connection as if it were initiated from the cloud services, using WebSocket binary frames. Please note that the figure only shows the most relevant packets. You may provide an access token by adding a query parameter to the WebSocket URI as defined by [RFC6750] Section 2.3.

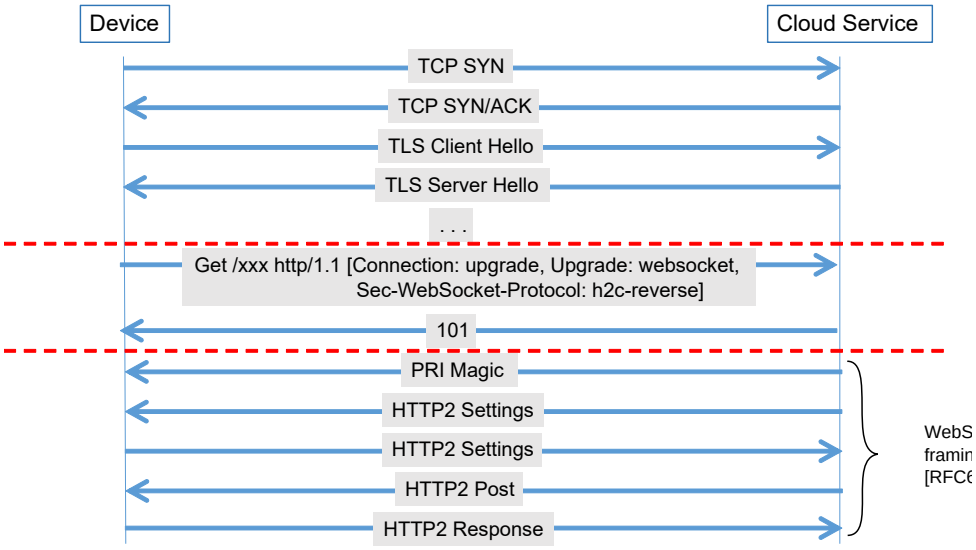

Figure 4: Connection initiation over websocket

This protocol shall be used when the uri protocol states 'wss'. A device signaling support for wss via its Protocol capability shall support h2c-reverse over WebSocket.

5.1.1.3 HTTP/2 Frames

Once an http/2 connection has been established the communication parameters are negotiated as specified by RFC 7540 so that the uplink can be used to exchange frames between the remote client and the local service.

5.1.1.4 HTTP Transactions

The uplink shall be used in reverse direction for http requests and responses. The remote client shall send requests that are served in a standard http manner by the local service.

5.2 Connection Management

A local service that is offered to a remote client for utilization is responsible for maintaining an operational communication channel. Since the connection needs to be established from the service to the remote client this connection is called uplink. The uplink shall be secured via TLS.

The local service shall monitor whether the remote client is able to communicate via the uplink. It may use the HTTP/2 ping mechanism to check whether a link is still operational if no packets have been received for a longer period of time.

The local service shall close and reconnect the uplink whenever no packets have been received from the remote client for more than 30 seconds. Each camera shall use an individual ascending interval strategy to avoid that all cameras connect at the same time.

The following example shows patterns chosen by two cameras A and B:

- Camera A: 3s, 6s, 12s, 24s, 30s, 30s, 30s ...
- Camera B: 2s, 4s, 8s, 16s, 30s, 30s, 30s ...

If multiple uplink configurations are included, the device shall try to establish all connections from the list and maintain those connections in parallel.

Note that this specification expects different clients may try to reach multiple local services and it is assumed services and clients are designed such that they do not interfere with each other. The coordination between such multiple clients and services is outside of the scope of the specification.

5.3 Authentication

Note that for the following discussion the roles of client and server are swapped.

The remote client shall authenticate itself using a valid server certificate. The local service shall verify the validity of the remote certificate according to RFC 6125.

5.3.1 mTLS authentication

When using mTLS authentication, the local service shall authenticate itself at the remote client using TLS client authentication according to RFC 5246 or subsequent specifications.

To uniquely identify a local service on remote client, it is recommended to have a unique client certificate installed on each local service. For example, CN field or Serial number of installed certificate could be used to uniquely identify the local service.

The device shall assign any tunneled HTTP or RTSP request the UserLevel that is configured for the configuration without the need of further authentication request headers.

If the `CertificateID` is present in the `UplinkConfiguration` the device shall use this authentication mechanism.

5.3.2 JWT authentication

When using JWT authentication, the local service shall provide a JWT token when sending the upgrade request.

The remote client shall validate the JWT token provided following the JWT specification as defined in the Core specification.

The device shall assign any tunneled HTTP or RTSP request the UserLevel that is configured for the configuration without the need of further authentication request headers.

When a device indicates JsonWebToken is supported through its Capabilities, then it shall support JWT authentication in this specification.

If the `AuthorizationServer` token is present in the `UplinkConfiguration` the device shall use this authentication mechanism.

5.4 Media Streaming

5.4.1 Introduction

The ONVIF Media Service Specification and the ONVIF Streaming Specification define various mechanism for streaming Video, Audio and Metadata. This specification defines how to apply streaming over http/2. Usage of other communication mechanisms like UDP unicast and multicast are outside of the scope of this specification.

5.4.2 RTSP over HTTP

The underlying RTSP over HTTP specification requires to open separate communication channels for continously sending and receiving data. For sending media from RTSP server to client the communication channel is opened with a GET request and a response without content-length. For sending media to the server a POST request is used. The specification notes that a content-length must be sent. Most applications use the value 32767 from the example given in the specification. Since http/2 streams may be closed by either side when the content-length has been reached clients shall use a value large enough for the connection life time.

5.4.3 RTSP over WebSockets

A device signaling support for websockets over RTSP via its capability RTSPWebSocketUri shall support RFC 8441 on the uplink.

6 Configuration Interface

6.1 Configuration parameters

- RemoteAddress : Uniform resource locator by which the remote client can be reached.
- CertificateID : ID of the certificate to be used for client authentication.
- UserLevel : Authorization level that will be assigned to the uplink connection
- Status : Current connection status
- CertPathValidationPolicyID : ID of Certificate Validation policy to validate the remote uplink server certficate. If not configured, server certificate shall not be validated.
- AuthorizationServer : JWTConfiguration token referring to an Authorization server that provides JWT token to authorize with the uplink server.
- Error : Optional user readable error information (readonly).

The device shall interpret the field RemoteAddress as key to a specific uplink configuration.

A device shall reject a configuration containing both CertificateID and AuthorizationServer.

6.2 GetUplinks

A device supporting uplinks shall support this command to retrieve the configured uplink configurations. The Status field shall signal whether a connection is Offline, Connecting or Online.

REQUEST:

This message is empty

RESPONSE:

- **Configuration – optional, unbounded [Configuration]**
  List of configurations.

ACCESS CLASS:

READ_SYSTEM

6.3 SetUplink

A device supporting uplinks shall support this command to add or modify an uplink configuration. The Status property of the UplinkConfiguration shall be ignored by the device. A device shall use the field RemoteAddress to decide whether to update an existing entry or create a new entry.

REQUEST:

- **Configuration – [UplinkConfiguration]**
  Configuration to be added or modified.

RESPONSE:

This message is empty

FAULTS:

- **env:Sender - ter:InvalidArgVal - ter:CertificateID**
  No certificate is stored under the requested CertificateID.
- **env:Sender - ter:InvalidArgVal - ter:CertPathValidationPolicyID**
  No certificate validation policy is stored under the requested CertPathValidationPolicyID.
- **env:Sender - ter:InvalidArgVal - ter:AuthorizationServerConfigurationToken**
  No AuthorizationServerConfiguration exists for the specified token.
- **env:Sender - ter:InvalidArgs - ter:MissingAuthentication**
  CertificateID and AuthorizationServerConfigurationToken are both missing.
- **env:Sender - ter:InvalidArgs - ter:AmbiguousAuthentication**
  CertificateID and AuthorizationServerConfigurationToken are mutually exclusive.
- **env:Sender - ter:InvalidArgs - ter:ProtocolNotSupported**
  The provided protocol is not supported by the device.

ACCESS CLASS:

WRITE_SYSTEM

6.4 DeleteUplink

A device supporting uplinks shall support this command to remove an uplink configuration.

REQUEST:

- **RemoteAddress – [xs:anyURI]**
  Configuration to be deleted.

RESPONSE:

This message is empty

FAULTS:

- **ter:Sender - ter:InvalidArgVal - ter:Address**
  No uplink exists for the given address.

ACCESS CLASS:

WRITE_SYSTEM

6.5 Capabilities

MaxUplinks	Maximum number of uplink connections that can be configured
Protocols	Supported protocols [https, wss]
AuthorizationModes	Supported authorization mode [mTLS, JWT]

Annex A.
Revision History

Rev.	Date	Editor	Changes
18.12	Dec 2018	Hans Busch	First release
22.06	June 2022	Hans Busch	Add section on media streaming
23.06	June 2023	Hans Busch	Clarify authentication within the established tunnel.
23.12	Dec 2023	Hans Busch	Add error codes.
24.06	June 2024	Jean-Francois Levesque	Add support for WebSocket protocol and token authorization.