ONVIF ™
Application Programmer's Guide

Version 1.0
May 2011

© 2011 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.

Table of Contents

Contributors

Johan Adolfsson	**Axis Communications AB**
Susanne Kinza	**Bosch Security Systems**
Daniel Fiala	**Dallmeier electronic GmbH & Co.KG**
Günther Frank	**Dallmeier electronic GmbH & Co.KG**
Takeshi Asahi	**Hitachi, Ltd.**
Hiroyuki Kanda	**Panasonic System Networks Co., Ltd.**
Hirokazu Kitaoka	**Panasonic System Networks Co., Ltd.**
Yohei Kushido	**Panasonic System Networks Co., Ltd.**
Scott Hudson	**Pelco by Schneider Electric**
Mike Kirby	**Pelco by Schneider Electric**
Kazunori Sakaki	**Sony Corporation**

1 Introduction

Open Network Video Interface Forum (ONVIF) is an open industry forum which was established in 2008 by Axis Communications, Bosch Security Systems, and Sony Corporation. It is committed to standardize communication between network devices to ensure interoperability between network products for the security market. Since its inception, ONVIF has published several documents and specifications describing and defining a flexible, scalable, and evolving interface that defines how security devices may be addressed and utilized. Along with its other activities, ONVIF seeks to provide a better and clearer understanding of the standard and its capabilities.

This document provides information about the use of the ONVIF standard from a programmer's perspective. It is intended as a complementary document to the ONVIF Core Specification [ONVIF] and the ONVIF Test Specification [TEST], and this document should not be considered as a standalone specification. For a strict definition of any of the technologies used or described in this document, refer to these two documents. Their contents overrule the descriptions and definitions found here.

This document is informative. Any normative documents have precedence over this document.

1.1 How to Use This Document

This book contains the following chapters and annexes:

- Chapter 1, Introduction
- Chapter 2, References
- Chapter 3, Abbreviations
- Chapter 4, Discovery
- Chapter 5, Initial Setup and Administration
- Chapter 6, Security
- Chapter 7, Streaming
- Chapter 8, Controlling
- Chapter 9, Eventing
- Chapter 10, Storage
- Chapter 11, Display
- Annex A, WSDL-Structures
- Annex B, SOAP Communication Traces from Use Case Examples
- Annex C, List of Functions with References
- Annex D, Pseudo Code Conventions

1.2 Conventions and Labels

The following typographic conventions are used in this document:

`Courier`	Indicates file names, command names, code samples, and onscreen output.
// Courier bold italic	Designates comments within code samples.
Italic	Used for emphasis, or as a substitute for an actual name or value. For example, the parameter *username* would be replaced by an actual user's name.

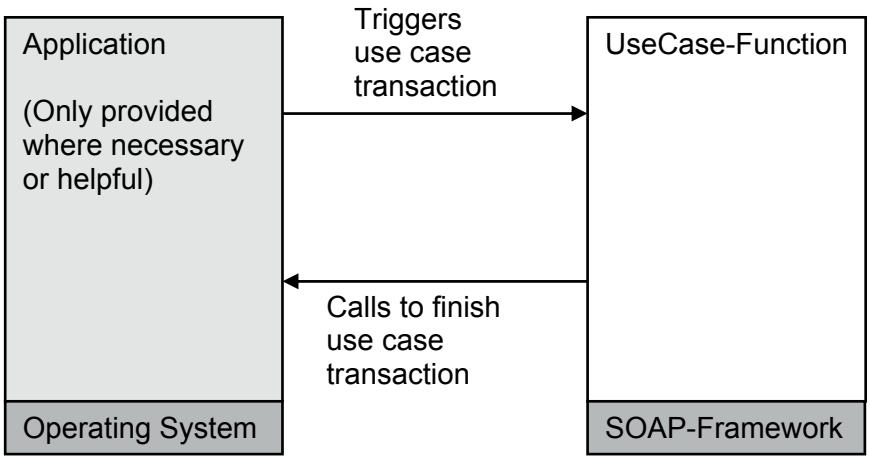
In addition, the following labels are used to indicate special types of information:

TIP:	Helpful, practical information that requires emphasis or does not otherwise fit into the flow of text.
NOTICE:	An explanation, comment, or a statement that is intended to catch the reader's attention.
 ***CAUTION:***	A potentially hazardous situation which, if not avoided, might result in minor or moderate injury or property damage. Risk of irreversible destruction to data or injury to a person. (This is not a life-threatening situation.)

1.3 Example Application Overview

In this document, each service description contains a source code (pseudo code) example. In these examples, an “Application” module named “App” is used to trigger all use case transactions. Each use case includes one or more methods, each with a self-explanatory name which can be referenced by other examples. This example method belongs to the module “ONVIF”.

The example applications are intended to guide the developer in a typical way of implementing a particular service or feature. The example application modules abstract what the client integrator must actually implement. Specifically, it abstracts the user, database, and other interactions.



1.4 Language Definition

This document uses pseudo code, a mixture of source code and English, to describe the algorithms. The pseudo code can easily be mapped to conventional scripting and programming languages.

Details that are not essential for human understanding of the algorithm are omitted or represented as calls to local methods of the `App` instance.

For a brief overview and definition of the language elements used, refer to Annex D, Pseudo Code Conventions.

2 References

[ONVIF]	ONVIF Core Specification Version 2.0, November 2010, available on http://www.onvif.org/imwp/download.asp?ContentID=19357
[ONVIF/Web Services framework:Security]	[ONVIF] Chapter 5.12, “Security,” starting on page 53
[ONVIF/Discovery]	[ONVIF] Chapter 7, “Device discovery,” starting on page 58
[ONVIF/Device management]	[ONVIF] Chapter 8, “Device management,” starting on page 69
[ONVIF/Backup]	[ONVIF] Chapter 8.3.3, “Backup,” starting on page 91
[ONVIF/Restore]	[ONVIF] Chapter 8.3.4, “Restore,” starting on page 92
[ONVIF/StartSystemRestore]	[ONVIF] Chapter 8.3.5, “Start system restore,” starting on page 92
[ONVIF/DeviceIO]	[ONVIF] Chapter 9, “Device IO Service,” starting on page 127
[ONVIF/Media]	[ONVIF] Chapter 11, “Media Configuration,” starting on page 147
[ONVIF/Realtime-Streaming]	[ONVIF] Chapter 12, “Realtime Streaming,” starting on page 197
[ONVIF/Example: Multicast Setup]	[ONVIF] Chapter 12.3.3.1, “Example: Multicast Setup,” starting on page 213
[ONVIF/Receiver-Configuration]	[ONVIF] Chapter 13, “Receiver Configuration,” starting on page 214
[ONVIF/Display-Service]	[ONVIF] Chapter 14, “Display Service,” starting on page 219
[ONVIF/Event-Handling]	[ONVIF] Chapter 15, “Event Handling,” starting on page 228
[ONVIF/PTZ]	[ONVIF] Chapter 16, “PTZ control,” starting on page 250
[ONVIF/Recording]	[ONVIF] Chapter 19, “Recording Control,” starting on page 303
[ONVIF/Search]	[ONVIF] Chapter 20, “Recording Search,” starting on page 319
[TEST]	ONVIF Test Specification Version 1.02.2, December 2010, available on http://www.onvif.org/imwp/download.asp?ContentID=19241
[devicemgmt.wsdl]	Official WSDL description file “Nov 2010 – ONVIF Device Management Service WSDL,” ver. 1.2, available on http://www.onvif.org/onvif/ver10/device/wsdl/devicemgmt.wsdl
[event.wsdl]	Official WSDL description file “Nov 2010 - ONVIF Event Service WSDL,” ver. 1.3, available on http://www.onvif.org/onvif/ver10/event/wsdl/event.wsdl

[media.wsdl]	Official WSDL description file “Nov 2010 - ONVIF Media Service WSDL,” ver. 1.2, available on http://www.onvif.org/onvif/ver10/media/wsdl/media.wsdl
[recording.wsdl]	Official WSDL description file “Nov 2010 - ONVIF Recording_Control Service WSDL,” ver. 1.0, available on http://www.onvif.org/onvif/ver10/recording.wsdl
[display.wsdl]	Official WSDL description file “Nov 2010 - ONVIF Display Service WSDL,” ver. 1.0, available on http://www.onvif.org/onvif/ver10/display.wsdl
[receiver.wsdl]	Official WSDL description file “Nov 2010 - ONVIF Receiver Service WSDL,” ver. 1.0, available on http://www.onvif.org/onvif/ver10/receiver.wsdl
[deviceio.wsdl]	Official WSDL description file “Nov 2010 - ONVIF Device_IO Service WSDL,” ver. 1.0, available on http://www.onvif.org/onvif/ver10/deviceio.wsdl
[ptz.wsdl]	Official WSDL description file “Nov 2010 - ONVIF PTZ Service WSDL,” ver. 2.0, available on http://www.onvif.org/onvif/ver20/ptz/wsdl/ptz.wsdl
[search.wsdl]	Official WSDL description file “Nov 2010 - ONVIF Recording_Search Service WSDL,” ver. 1.0, available on http://www.onvif.org/onvif/ver10/search.wsdl
[onvif.xsd]	Official schema description file “Nov 2010 - ONVIF Schema,” ver. 1.2, available on http://www.onvif.org/onvif/ver10/schema/onvif.xsd
[WS-Addressing]	http://www.w3.org/Submission/ws-addressing/
[WS-BaseNotification]	http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf
[WS-Discovery]	“Web Services Dynamic Discovery (WS-Discovery),” J. Beatty et al., April 2005. Available at http://specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf
[DHCP]	IETF RFC 2131, Dynamic Host Configuration Protocol, http://www.ietf.org/rfc/rfc2131.txt
[NTP]	IETF RFC 5905, Network Time Protocol Version 4: Protocol and Algorithms Specification, http://www.ietf.org/rfc/rfc5905.txt
[MTOM]	SOAP Message Transmission Optimization Mechanism, http://www.w3.org/TR/soap12-mtom/
[HTTP]	Hypertext Transfer Protocol -- HTTP/1.1, http://tools.ietf.org/html/rfc2616
[RTSP]	IETF RFC 2326, Real Time Streaming Protocol (RTSP), http://www.ietf.org/rfc/rfc2326.txt
[SDP]	IETF RFC 4566, SDP: Session Description Protocol, http://www.ietf.org/rfc/rfc4566.txt

3 Abbreviations

This document contains the following abbreviations:

CA: Certificate Authority

DHCP: Dynamic Host Configuration Protocol

IPv4: Internet Protocol version 4

LAN: Local Area Network

MTOM: Message Transmission Optimization Mechanism - A method of efficiently sending binary data (usually using Base 64 encoding) to and from Web services. MTOM is usually used with XML-binary Optimized Packaging (XOP).

NTP: Network Time Protocol

NVT: Network Video Transmitter – A device which is capable of delivering live streams.

ONVIF: Open Network Video Interface Forum

PKCS: Public-Key Cryptography Standards

PTZ: Pan/Tilt/Zoom

RTSP: Realtime Streaming Protocol

SDP: Session Description Protocol

SOAP: Simple Object Access Protocol

TLS: Transport Layer Security

UDP: User Datagram Protocol

WS: Web Services

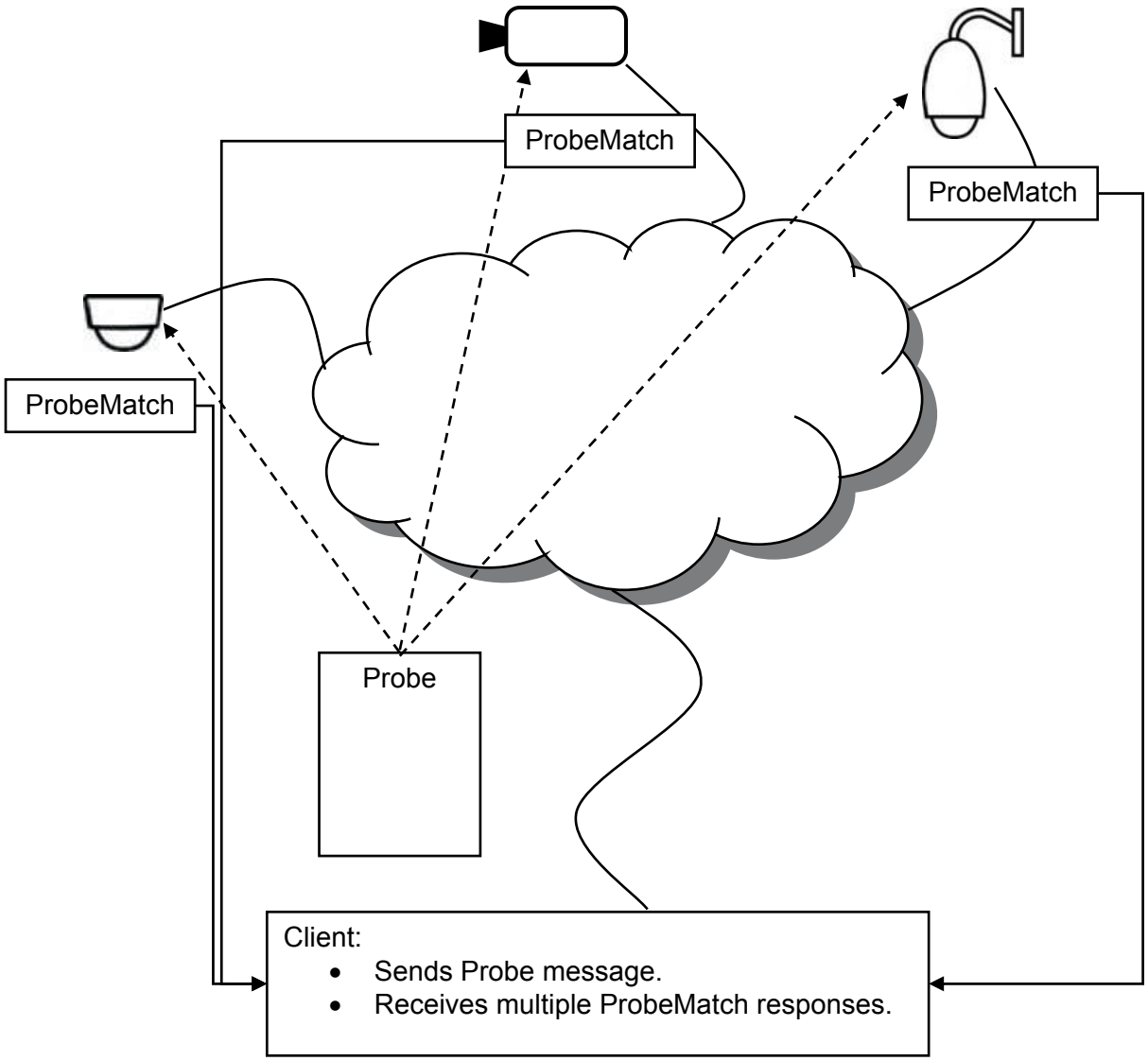
WSDL: Web Services Description Language – A scheme to describe server methods and their parameters for remote invocation.

4 Discovery

ONVIF devices support WS-Discovery, which is a mechanism that supports probing a network to find ONVIF capable devices. For example, it enables devices to send `Hello` messages when they come online to let other devices know they are there. In addition, clients can send `Probe` messages to find other devices and services on the network. Devices can also send `Bye` messages to indicate they are leaving the network and going offline.

Messages are sent over UDP to a standardized multicast address and UDP port number. All the devices that match the types and scopes specified in the `Probe` message respond by sending `ProbeMatch` messages back to the sender.

WS-Discovery is normally limited by the network segmentation at a site since the multicast packages typically do not traverse routers. Using a Discovery Proxy could solve that problem, but details about this topic are beyond the scope of this document. For more information, see [ONVIF/Discovery] and [WS-Discovery].


ProbeMatch
ProbeMatch
ProbeMatch
Probe
Client:
• Sends Probe message.
• Receives multiple ProbeMatch responses.

4.1 Prerequisites

None.

4.2 Targeted Services and Technologies

- [WS-Discovery]
- [ONVIF/Device] and [devicemgmt.wsdl]

4.3 ONVIF::Discovery

In the Discovery use case, we send a WS-Discovery `Probe` message and wait for `ProbeMatch` responses. The responses are processed, and relevant info is stored in a list for processing later, as shown in Section 5.1.3, ONVIF::ProcessMatch.

```
// Send WS-Discovery Probe, collect the responses and then
// process the responses.
probematch_type probematcheslist[]={};
// Send probe. See chapter 4.3.1 for details
probe = ONVIF::DiscoverySendProbe(scopes, types);

// Wait a while for responses
while (data_available_and_not_timeout(probe.net_handle))
{
    // This fetch next probe match so that we can put it into the list
    // See chapter 4.3.2 for details
    probematch = ONVIF::DiscoveryReadResponse(probe);
    // Store info about the match, first check for duplicates
    if (!in_list(probematcheslist, probematch))
    {
        add_to_list(probematcheslist, probematch);
    }
}

// Process the responses, see chapter 5.1.3 for details.
foreach (probeMatch in probematcheslist)
{
    ONVIF::ProcessMatch(probeMatch);
}
```

TIP: To maximize the number of discovered devices, best practice would be to collect all available responses before processing the contents of each individual response message. Responses may be lost if too much time is spent processing individual responses during the brief time that all the devices on the network are responding to the probe.

4.3.1 ONVIF::DiscoverySendProbe

This function composes and sends a WS-Discovery `Probe` for the specified scopes and types. More information regarding scopes and types can be found in the ONVIF Core Specification [ONVIF/Discovery] and the referenced WS-Discovery specification [WS-Discovery].

Input parameters are:

- `scopes` – The type of services, location, hardware, name, and so on to discover [ONVIF Scopes section 7.3.3.2]
- `types` – The device type to discover, for example, `tds:Device`, `dn:NetworkVideoTransmitter` [ONVIF 4.4, 7.3.3.1]

```
DiscoverySendProbe(scopes, types)
{
  // Each probe should have a unique MessageID to be able to match
  // requests and responses. We store it in the probe place holder for later checking
  probe.MessageID = "uuid:" + App.uuidGenerate();

  // Build probe message, we provide
  // MessageID, Types and Scopes. See SOAP trace section for how
  // it actually looks like.

  message = App.BuildProbeMessage(probe.MessageID, types, scopes);

  // Send probe to multicast address and port according to [WS-Discovery] section 2.4
  probe.net_handle = App.send_multicast("239.255.255.250", 3702, message);
  return probe;
}
```

4.3.2 ONVIF::DiscoveryReadResponse

This function reads and processes responses to `Probe` messages, then updates `probematcheslist`.

```
DiscoveryReadResponse(probe)
{
  // Read response and process it.
  // We need both the body and the Header:
  aProbeMatches = App.ReadProbeMatches(probe.net_handle, Header);

  // Check if this is a response to the probe we sent:
  if (Header.RelatesTo != probe.MessageID) {
    return -1;
  }
  // We pick what we need from the response:
  probematch.types = aProbeMatches.ProbeMatch.Types;
  probematch.scopes = aProbeMatches.ProbeMatch.Scopes;
  // XAddrs is a space separated list of URLs to the Device service:
  probematch.xaddrs = aProbeMatches.ProbeMatch.XAddrs;
  probematch.metadataversion = aProbeMatches.ProbeMatch.MetadataVersion;
  probematch.address = aProbeMatches.ProbeMatch.EndpointReference.Address;

  return probematch;
}
```

5 Initial Setup and Administration

5.1 First Actions After Discovery

After ONVIF devices are discovered using WS-Discovery, you would typically access a device using the supplied `XAddrs` to test where it is reachable. Use `device.GetSystemDateAndTime` to accomplish this because it should not require authentication. You can also consider calling `device.GetDeviceInformation` and `device.GetCapabilities`.

5.1.1 Prerequisites

- Devices must already be discovered.
- At least one valid `XAddrs` for the device service entry point.

5.1.2 Targeted Services and Technologies

- [ONVIF/Device] and [devicemgmt.wsdl]

5.1.3 ONVIF::ProcessMatch

When processing the list of discovered devices, first do a `device.GetDeviceSystemDateAndTime()` call:

- To determine if the device is reachable with the supplied `XAddr`
- To obtain the device time
- To determine the time offset, which will be used later when authenticating (see Section 6.1, Authentication, for more information)

The following subfunction describes how a `ProbeMatch` message should be processed. It takes the following parameter:

- `aProbeMatch` - A `ProbeMatch` message received as reaction to a `Probe` request.

```
// Try to connect to the device and store the XAddr that works.
// Use GetSystemDateAndTime for checking and to get the time offset.
// Find out information and capabilities etc.
ProcessMatch(aProbeMatch)
{
   // For authentication to work, it is important to have
   // the same time in device and client so the Created time
   // is not off with too much. E.g. 5 seconds.
   // device.GetSystemDateAndTime should not require authentication,
   // we use that function to check the XAddrs in the ProbeMatch response.
   // onvifdev is a placeholder that we populate with valid XAddr,
   // authentication info and capabilities etc.
   onvifdev = App.createNewDeviceInstance();
   device = null;

   systemDateAndTime = ONVIF::CheckXaddrsAndGetTime(device, onvifdev, aProbeMatch);
   if (!aSystemDateAndTime) {
     // No valid response - probably communication or authentication failure!
     // Return and try next XAddr
     return;
   }

   // Generate a time offset based on device time
   // Either UTCDateTime or LocalDateTime or both should be present.
   devicetime = ParseSystemDateAndTime(aSystemDateAndTime);
```

```
    // Store time offset in the device place holder for later use
    // when creating service objects and handling authentication.
    onvifdev.timeoffset = devicetime - time(NULL) - 1;

    // Assign desired username and password to be used
    // (framework dependant). See chapter 6.1 for details.
    App.SetAuthenticationInformation(onvifdev);
    ONVIF::UpdateCapabilities(device ,onvifdev);
 }
```

5.1.4 ONVIF::UpdateCapabilities

After you have established a valid way to communicate with the device, use actions such as `GetDeviceInformation`, `GetCapabilities`, and possibly `GetSystemLog` to obtain state and capabilities for the device. You also must obtain the URIs for the various services that the device supports.

Parameters are:

- `device` – An instance that represents the device service.
- `onvifdev` – Data object for collecting and combining device-specific information that is required for various services. For example, storing service URLs that are required for media, PTZ, or other services.

```
UpdateCapabilities(device, onvifdev)
{
   aDeviceInformation = device.GetDeviceInformation();

   // Store/Process interesting stuff from DeviceInformation
   onvifdev.model  = aDeviceInformation.Model;
   onvifdev.serial = aDeviceInformation.SerialNumber;

   // Get the capabilities and
   // store all or interesting portions of the capabilities.
   onvifdev.capabilities = device.GetCapabilities(Category = "All");

   // Store the XAddr of the different services - these will be used
   // later on when accessing Events and Media service.
   onvifdev.devicexaddr = onvifdev.capabilities.Device.XAddr;
   onvifdev.eventsxaddr = onvifdev.capabilities.Events.XAddr;
   onvifdev.mediaxaddr  = onvifdev.capabilities.Media.XAddr;
   // Other and future ONVIF devices may have additional services

   // If Capabilities/System/SystemLogging is true,
   // the log could be interesting.
   if (aProbeMatch.capabilities.System.SystemLogging) {
     log = device.GetSystemLog(LogType = "System");
     App.CheckLog(onvifdev, log);
   }
}
```

NOTICE: Service addresses are not fixed. They vary according to the vendor and device, and might require different handling. Therefore, this information must be updated dynamically.

5.1.5 ONVIF::CheckXAddrsAndGetTime

This function connects to the device and calls `GetSystemDateAndTime` using the `XAddrs` list in `aProbeMatch`. Store the successful `XAddr` in the `onvifdev` placeholder.

Parameters are:

- `device` – An instance that represents the device service.
- `onvifdev` – Data object for collecting and combining device-specific information that is required for various services. For example, storing service URLs that are required for media, PTZ, or other services.
- `aProbeMatch` – A `ProbeMatch` message received as reaction to a `Probe` request.

```
SystemDateAndTime CheckXaddrsAndGetTime(device,onvifdev, aProbeMatch)
{
   // Check all the XAddr in the XAddrs returned in the ProbeMatch
   // and save the first one we can use.
   onvifdev.xaddr = first(aProbeMatch.xaddrs);

   while (onvifdev.xaddr) {
     // here you see an example how the internals of a getDeviceService() method
     // might work out.
     device = App.createEnpointForDeviceService(onvifdev.devicexaddr);
     aSystemDateAndTime = device.GetSystemDateAndTime();
     // if we successfully connected to the device and got some valid answer
     if (present(aSystemDateAndTime)) {
       return aSystemDateAndTime;
     }
     onvifdef.xaddr = next(aProbeMatch.xaddrs);
   }
   return null; // No XAddr that works for us!
}
```

5.1.6 ONVIF::ParseSystemDateAndTime

This function parses a `SystemDateAndTime` response from the `GetSystemDateAndTimeResponse` message and returns the time. This function demonstrates the use of both `LocalDateTime` and `UTCDateTime`.

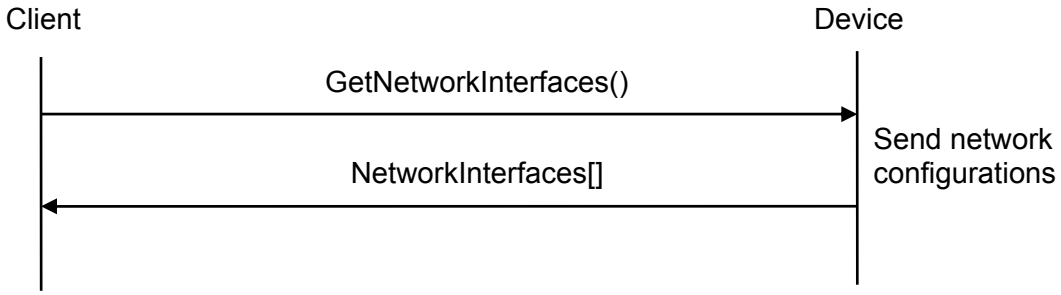
Parameters are:

- `aSystemDateAndTime` – Response object to a corresponding `GetSystemDateAndTime` request

```
time_t ParseSystemDateAndTime(aSystemDateAndTime)
{
   // Parse out the device time from aSystemDateAndTime.
   // Either UTCDateTime or LocalDateTime or both should be present.
   // From ONVIF 2.0 UTCDateTime is mandatory, but before that it was not.
   // Lets handle both cases so that we can get the LocalTime from old devices
   // or we may fail to authenticate and not be able to e.g. upgrade firmware.

   tm tmrdev;
   if (aSystemDateAndTime.UTCDateTime) {
     is_utc = true;
     aDateTime = aSystemDateAndTime.UTCDateTime;
   } else {
     is_utc = false;
     aDateTime = aSystemDateAndTime.LocalDateTime;
   }
   tmrdev.tm_mday = aDateTime.Date.Day;
   tmrdev.tm_mon  = aDateTime.Date.Month - 1;
   tmrdev.tm_year = aDateTime.Date.Year - 1900;
   tmrdev.tm_sec  = aDateTime.Time.Second;
   tmrdev.tm_min  = aDateTime.Time.Minute;
   tmrdev.tm_hour = aDateTime.Time.Hour;

   // Return the time provided by the device so that we can use it for
   // authentication and store time offset in the device place holder for later use
   if (is_utc) {
     return mktime(tmrdev) - timezone;
   } else {
     return mktime(tmrdev);
   }
 }
```

5.2 Getting the Network Interface Configuration

Part of the basic device configuration is the network setup. This use case shows how to obtain the network interface configurations from the device.



5.2.1 Prerequisites

None.

5.2.2 Targeted Services and Technologies

- [ONVIF/Device] and [devicemgmt.wsdl]

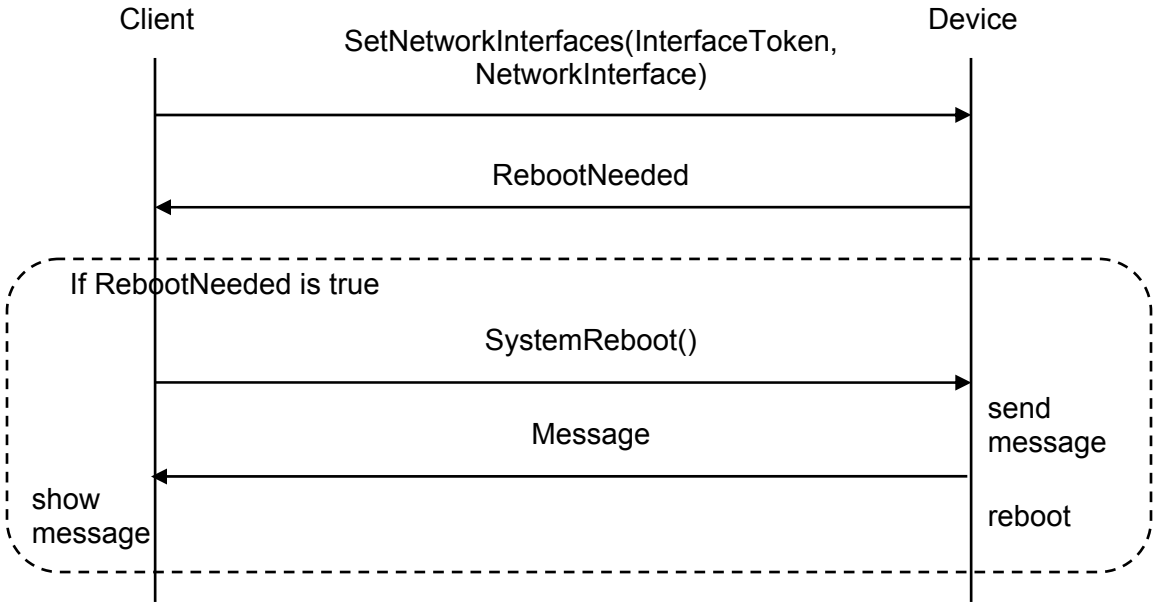
5.2.3 ONVIF::GetNetworkInterfaceConfiguration

This use case shows how to obtain the IPv4 address of the network interface device, for example, `eth0`.

```
// step 1 - create the device service object
//          and get the network interface configurations of the device
deviceService = getDeviceService( myDeviceServiceAddress );
// See Annex B for soap trace
networkInterfaceList = deviceService.GetNetworkInterfaces();

//step 2 - find the configuration of "eth0"
eth0 = null;
foreach( networkInterface in networkInterfaceList )
{
    //search the configuration of "eth0" in networkInterfaceList
    if ( present( networkInterface.Info ) &&
         present( networkInterface.Info.Name ) &&
         networkInterface.Info.Name == "eth0" )
    {
        //discovered the configuration of "eth0" in networkInterfaceList
        eth0 = networkInterface;
        break;
    }
}
//step 3 - show the IP(s) in configuration of "eth0"
if ( present ( eth0 ) )
{
    //show the IPv4 address settings, if present
    if( present(eth0.IPv4 ) )
    {
        if( present(eth0.IPv4.Config.Manual ) )
        {
            //list of manual IPv4 addresses and prefix lengths
            foreach( prefixedIPv4Address in eth0.IPv4.Config.Manual )
            {
                address = prefixedIPv4Address.Address;
                prefixLength = prefixedIPv4Address.PrefixLength;
                App.printIPAddress( address, prefixLength ); //just show the IP
            }
        }
        if( present(eth0.IPv4.Config.LinkLocal ) )
        {
            //link-local IPv4 address and prefix length
            address = eth0.IPv4.Config.LinkLocal.Address;
            prefixLength = eth0.IPv4.Config.LinkLocal.PrefixLength;
            App.printIPAddress( address, prefixLength ); //just show the IP
        }
        if( present( networkInterface.IPv4.Config.FromDHCP ) )
        {
            //IPv4 address and prefix length assigned from DHCP
            address = networkInterface.IPv4.Config.FromDHCP.Address;
            prefixLength = networkInterface.IPv4.Config.FromDHCP.PrefixLength;
            App.printIPAddress( address, prefixLength ); //just show the IP
        }
        //show the DHCP enable flag on the screen
        App.printDHCPEnableFlag( networkInterface.IPv4.Config.DHCP );
    }
}
```

5.3 Setting Network Interface Configuration

Managing the device might require changing the network interface configuration of the device. This use case shows how to set the network interface configurations to the device.



5.3.1 Prerequisites

None.

5.3.2 Targeted Services and Technologies

- [ONVIF/Device] and [devicemgmt.wsdl]

5.3.3 ONVIF::SetNetworkInterfaceConfiguration

This example shows how to set a new IPv4 address to the network interface `eth0` of the device. It attempts to set `192.168.0.200` with the netmask `255.255.255.0` as a static IP configuration.

```
//create the device service object
deviceService = getDeviceService( myDeviceServiceAddress );

//create the parameter object of the SetNetworkInterfaces
interfaceToken = "eth0"; //select or set the interface token

networkInterfaceSetConfiguration.Enabled = true; //enable this new configuration
networkInterfaceSetConfiguration.IPv4.Enabled = true; //enable it to be IPv4
networkInterfaceSetConfiguration.IPv4.Manual.Address = "192.168.0.200"; //ip address
networkInterfaceSetConfiguration.IPv4.Manual.PrefixLength = 24; //netmask
networkInterfaceSetConfiguration.IPv4.DHCP = false; // static configuration

//invoke SetNetworkInterfaces to set the network interface
//configurations to the device
// See Annex B for soap trace
rebootNeeded = dev.SetNetworkInterfaces( interfaceToken,
                                networkInterfaceSetConfiguration );

//reboot the device if necessary
```

```
if( rebootNeeded )
{
    // See Annex B for soap trace
    message = SystemReboot();

    //show the reboot message on the screen
    App.printRebootMessage( message );
}

//if the client communicates with the device which got a new IP address,
//change the referring IP address at the  client side if necessary
```

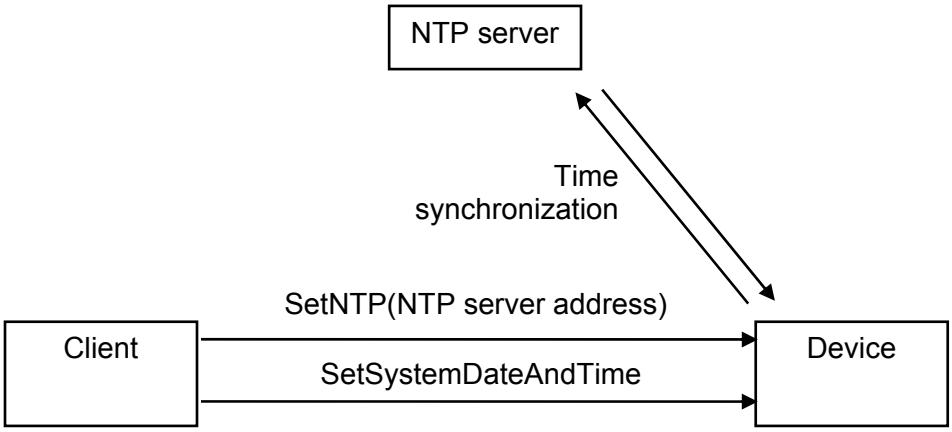
5.4 Time Synchronization Including NTP Configuration (Set Manually)

For a device to operate properly, it must be set with the correct time. This use case shows one way to synchronize the clock of the device with an NTP server.

The device can acquire the NTP server address in one of the following ways:

- The client directly sends `SetNTP()` including the NTP server address.
- The NTP server address is acquired from a DHCP server.

This section describes the first scenario. (The following section describes the second scenario.)



5.4.1 Prerequisites

Assume that the NTP server address is `192.168.10.1`.

5.4.2 Targeted Services and Technologies

- [ONVIF/Device] and [devicemgmt.wsdl]

5.4.3 ONVIF::TimeSynchronizationWithNTPServerSetByManual

This use case shows one way to set up time synchronization between the device and the NTP server.

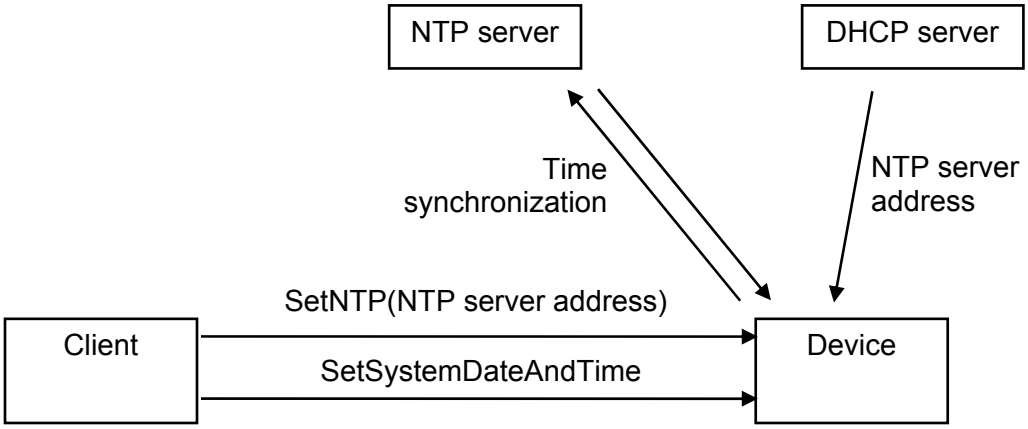
In step 1, the client obtains the current system time setting of the device. In step 2, if the device is not already using the NTP server, the client sets the NTP server address to the device. In step 3, the client invokes `SetSystemDateAndTime()` to the device. This results in time synchronization between the device and the NTP server.

```
//create the device service object
deviceService = getDeviceService( myDeviceServiceAddress );

//step 1 - get current system time setting
// See Annex B for soap trace
systemDateAndTime = deviceService.GetSystemDateAndTime();

//step 2 - set NTP server address, if NTP is not in use
if( systemDateAndTime.DateTimeType != "NTP" )
{
    //set NTP server address
```

```
    fromDHCP = false;
    ntpServerList[0].Type = "IPv4";
    ntpServerList[0].IPv4Address.Address = "192.168.10.1";
    // See Annex B for soap trace
    deviceService.SetNTP( fromDHCP, ntpServerList );
}

//step 3 - set system date and time of NVT using NTP
dateTimeType = "NTP";
daylightSavings = false;

// See Annex B for soap trace
deviceService.SetSystemDateAndTime( dateTimeType, daylightSavings );
```

5.5 Time Synchronization Including NTP Configuration (Set by DHCP)

As discussed in the previous section, for a device to operate properly, it must be set with the correct time. This use case describes how to synchronize the clock of the device by acquiring the NTP server address from a DHCP server. (The previous section describes how the time can be set manually, when a client directly sends the NTP server address to the device.)



5.5.1 Prerequisites

- Assume that the NTP server address is `192.168.10.1`.
- A DHCP server must be enabled.

5.5.2 Targeted Services and Technologies

- [ONVIF/Device] and [devicemgmt.wsdl]

5.5.3 ONVIF::TimeSynchronizationWithNTPServerSetByDHCP

This use case shows another way to set up time synchronization between the device and the NTP server.

In step 1, the client obtains the current system time setting of the device. In step 2, if the device is not already using the NTP server, the client sets the device to acquire the NTP server address from DHCP. In step 3, the client invokes `SetSystemDateAndTime()` to the device. This results in time synchronization between the device and the NTP server.

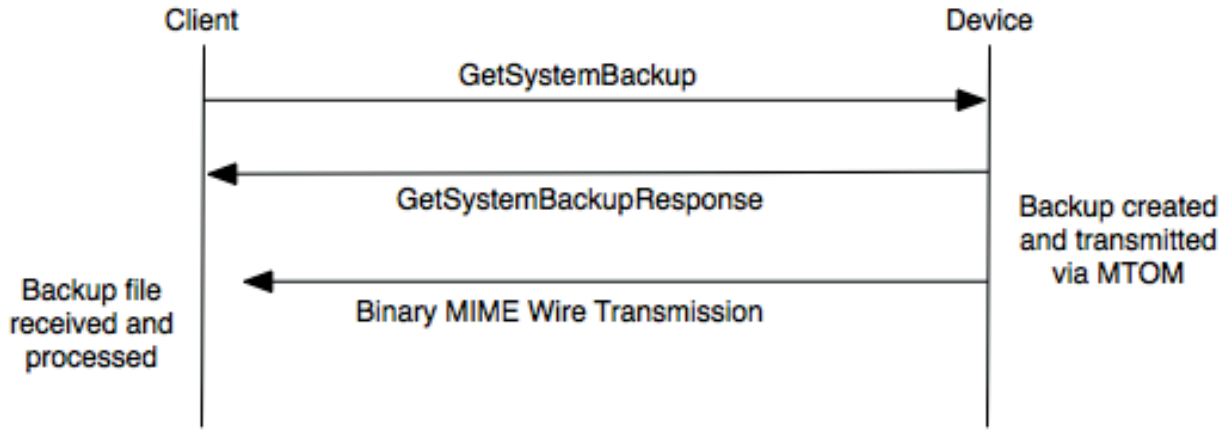
```
//create the device service object
deviceService = getDeviceService( myDeviceServiceAddress );

//step 1 - get current time setting
// See Annex B for soap trace
systemDateAndTime = deviceService.GetSystemDateAndTime();

//step 2 - set NVT to use DHCP to acquire NTP server addresses, if NTP is not in use
if( systemDateAndTime.DateTimeType != "NTP" )
{
    fromDHCP = true;
    nTPManual = null;
    // See Annex B for soap trace
    deviceService.SetNTP( fromDHCP, nTPManual );
}
```

```
//step 3 - set system date and time of NVT using NTP
dateTimeType = "NTP";
daylightSavings = false;
deviceService.SetSystemDateAndTime( dateTimeType, daylightSavings );
```

5.6 Backup System Configuration Files from a Device

This operation retrieves one or more system backup configuration files from a device. The device should support the return of backup configuration files through the `GetSystemBackup` command. Backup files are returned with reference to a name and MIME-type, together with binary data. The backup configuration files are transmitted through MTOM.

NOTICE: The format and structure of the backup file(s) is outside the scope of ONVIF and is not defined in the current specification.



5.6.1 Prerequisites

None.

5.6.2 Targeted Services and Technologies

- System Backup [ONVIF/Backup]
- Device Service [devicemgmt.wsdl]
- MTOM [MTOM]

5.6.3 ONVIF::GetSystemBackupRequest

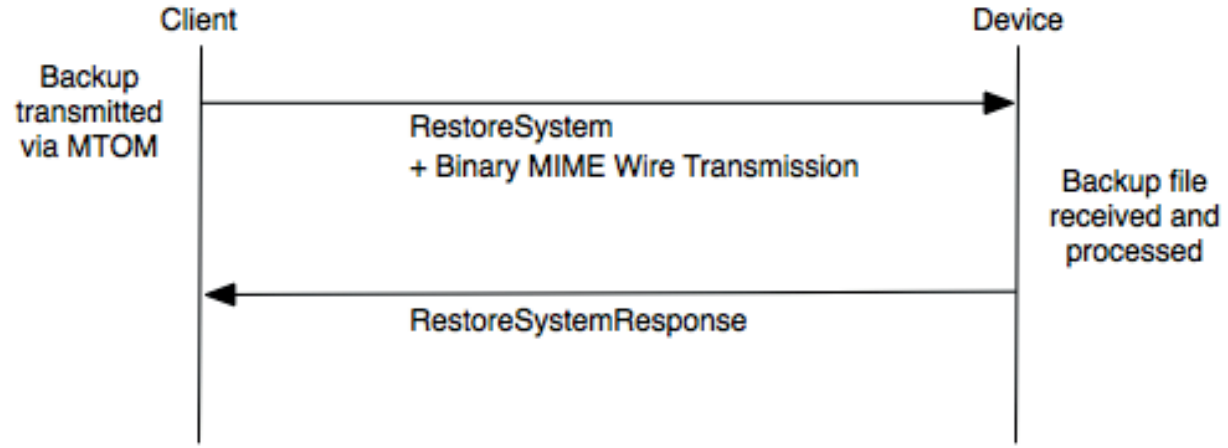
First, a backup request is generated and sent to the NVT (`GetSystemBackup`). The NVT responds with a reference to the file name, MIME type, and binary data (using MTOM).

```
//get device service end point and send GetSystemBackup request which returns
// a list of backup files
deviceService = getDeviceService(MyDeviceServiceAddress);
BackupFileList = deviceService.GetSystemBackup();

//Iterate over each backup file returned
foreach (backupfile in BackupFileList)
{
    //Use the name and data to write to disk (outside the scope of this document)
    App.saveBackupFile(backupfile.Name, backupfile.Data);
}
```

5.7 Restore System Configuration Files to a Device

This operation restores system backup configuration files that were previously retrieved from a device. The device should support the restore operation through the `RestoreSystem` command. If this command is supported, the device can accept backup files returned by the `GetSystemBackup` command. The backup configuration files are transmitted through MTOM.

NOTICE: The format and structure of the backup file(s) is outside the scope of ONVIF and is not defined in the current specification.



5.7.1 Prerequisites

A backup file must be available.

5.7.2 Targeted Services and Technologies

- System Restore [ONVIF/Restore]
- Device Service [devicemgmt.wsdl]
- MTOM [MTOM]

5.7.3 ONVIF::RestoreSystem

First, a `RestoreSystem` request is sent to the NVT along with the system backup configuration file (through MTOM). If the request is successful, the NVT responds with an empty message. If the request is not successful, a fault code is returned: `env:Sender`, `ter:InvalidArgVal`, `ter:InvalidBackupFile (The backup file(s) are invalid.)`

```
//Restore 2 files to NVT, service takes 1 or more files
filenameList[0] = "system_settings.zip";
filenameList[1] = "user_settings.zip";

//Iterate over the named files
foreach (filename in filenameList)
{
    //BackupFile is read from disk using a helper function
    //(outside scope of this document)
    BackupFile = readBackupFile(filename);

    //Add the backup file to the BackupFile array
    BackupFiles.add(BackupFile);
}

deviceService = getDeviceService(MyDeviceServiceAddress);

//Send the BackupFile array to the NVT
// See Annex B for soap trace
deviceService.RestoreSystem(BackupFiles);
```

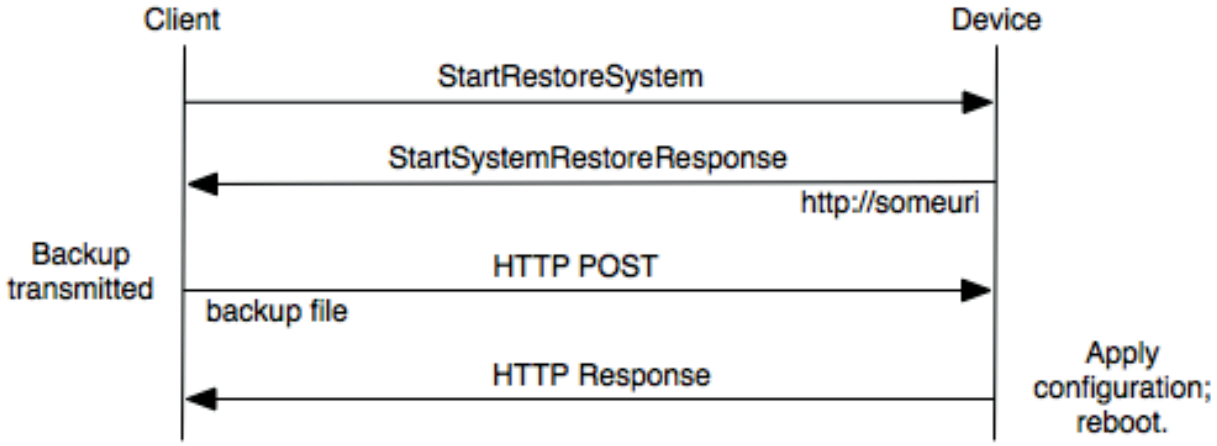
5.8 Start System Restore via HTTP Post

This method is the easiest way to restore a system.

This operation initiates a system restore from backed up configuration data using the HTTP POST mechanism. The response to the command includes an HTTP URL where the backup file may be uploaded.

NOTICE: The format and structure of the backup file(s) is outside the scope of ONVIF and is not defined in the current specification.

The actual restore takes place as soon as the HTTP POST operation has completed. Devices should support system restore through the `StartSystemRestore` command.



5.8.1 Prerequisites

A backup file must be available.

5.8.2 Targeted Services and Technologies

- Start System Restore [ONVIF/StartSystemRestore]
- Device Services [devicemgmt.wsdl]
- HTTP [HTTP]

5.8.3 ONVIF::StartSystemRestore

System restore over HTTP involves the following steps:

1. The client calls `StartSystemRestore`.
2. The device responds with the upload URI.
3. The client transmits the configuration data to the upload URI using HTTP POST.
4. The device applies the uploaded configuration, then reboots if necessary.

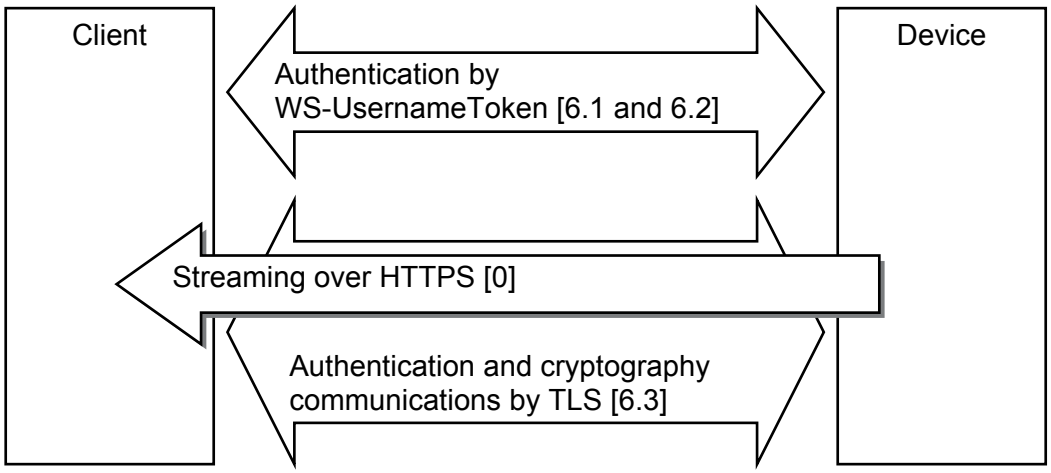
If the system restore fails because the uploaded file was invalid, the HTTP POST response is `415 Unsupported Media Type`. If the system restore fails due to an error at the device, the HTTP POST response is `500 Internal Server Error`.

The value of the Content-Type header in the HTTP POST request is `application/octet-stream`.

```
filename = "system_settings.zip"
BackupFile = readBackupFile(filename);

deviceService = getDeviceService(MyDeviceServiceAddress);

//Request NVT to start system restore, response contains URL to post backup file
response = deviceService.StartSystemRestore();

//Get a helper client (outside the scope of the spec)
httpSystemRestoreClient = App.getHttpSystemRestoreClient(MyDeviceServiceAddress);

//Send the backup file using http post to the uri returned from the NVT's
StartSystemRestore response
httpSystemRestoreClient.send(response.UploadUri, BackupFile.Name, BackupFile.Data);
```

6 Security

This chapter describes how to use the security function included in the ONVIF specification. It covers authenticating by `WS-UsernameToken`, communicating by TLS, and streaming over HTTPS.



6.1 Authentication

When a device requires authentication to access a web service, the client uses `WS-UsernameToken` for the device. The ONVIF specification does not include an example that shows how `WS-UsernameToken` works, so this chapter describes how to establish authentication between a client and a device.

6.1.1 Authenticating by WS-UsernameToken

This chapter describes how to add the `WS-UsernameToken` element to the SOAP header when a device requires authentication.

6.1.1.1 Prerequisites

- The device requires authentication by `WS-UsernameToken`.
- The user with appropriate authority is already registered with the device.

6.1.1.2 Targeted Services and Technologies

- `WS-UsernameToken` [ONVIF/Web Services framework:Security]

6.1.1.3 ONVIF::AuthenticatingByWS-UsernameToken

When a device requires authentication through `WS-UsernameToken`, the client must set user information with the appropriate privileges in `WS-UsernameToken`. This use case contains an example of setting that user information using `SetHostname`.

`WS-UsernameToken` requires the following parameters:

- `Username` – The user name for a certified user.
- `Nonce` – A random, unique number generated by a client.
- `Created` – The `UtcTime` when the request is made.
- `Password` – The password for a certified user. According to the ONVIF specification, `Password` should not be set in plain text. Setting a password generates `PasswordDigest`, a digest that is calculated according to an algorithm defined in the specification for `WS-UsernameToken`:

  Digest = `B64ENCODE( SHA1( B64DECODE(` ***Nonce*** `)` + ***Date*** + ***Password*** `) )`

For example:

- ***Nonce*** – `LKqI6G/AikKCQrN0zqZFlg==`
- ***Date*** – `2010-09-16T07:50:45Z`
- ***Password*** – `userpassword`
- Resulting ***Digest*** – `tuOSpGlFlIXsozq4HFNeeGeFLEI=`

Process:

1. The client sets the user with appropriate privileges in `WS-UsernameToken`. Four parameters are required, as described below.
2. The client sends the `SetHostname` to the device with `WS-UsernameToken`.

```
// create the device object to use the service
deviceService = getDeviceService(MyDeviceServiceAddress)

// The user name of a certified user is set
wsUsernameToken.Username = "username";

// A random number value generated with a client uniquely is set
nonceBinaryData = getNonce();
nonceBase64 = base64(nonceBinaryData);
wsUsernameToken.Nonce = nonceBase64;

// The time at the time of the request making is set
utctimeData = getUTCTime();
utctimeStringData = uTCTime2DatetimeString(utctimeData);
utctimeBinaryData = string2Binary(utctimeStringData);
wsUsernameToken.Created = utctimeStringData;

// The password digest of a certified user is set
password = "userpassword";
passwordBinaryData = string2Binary(password);
passwordDigest = SHA1(nonceBinaryData + utctimeBinaryData + passwordBinaryData);
passwordDigestBase64 = base64(passwordDigest);
wsUsernameToken.Password = passwordDigestBase64;
wsUsernameToken.PasswordType = "http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-username-token-profile-1.0#PasswordDigest";

// The client sends the SetHostname request to the device with WS-UsernameToken
hostname = "camera1";
deviceService.SoapHeader.WsUsernameToken = wsUsernameToken;
try
{
    deviceService.SetHostname(hostname);
} catch (SOAPFaultException e)
{
    // If the device does not accept specified username and password, the client
    // required to change user information.
    if(e.subcode == NotAuthorized)
    {
        App.changingUserInformation();
    }
}
```

6.1.1.4 SOAP Communication Trace

6.1.1.4.1 WS-UsernameToken

Request **SetHostname** with **WS-UsernameToken**

```
<?xml version="1.0" encoding="utf-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd"
xmlns:tds="http://www.onvif.org/ver10/device/wsdl">
  <SOAP-ENV:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>username</wsse:Username>
```

```
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-
1.0#PasswordDigest">tuOSpGlFlIXsozq4HFNeeGeFLEI=</wsse:Password>
        <wsse:Nonce>LKqI6G/AikKCQrN0zqZFlg==</wsse:Nonce>
        <wsu:Created>2010-09-16T07:50:45Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tds:SetHostname>
      <tds:Name>camera1</tds:Name>
    </tds:SetHostname>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

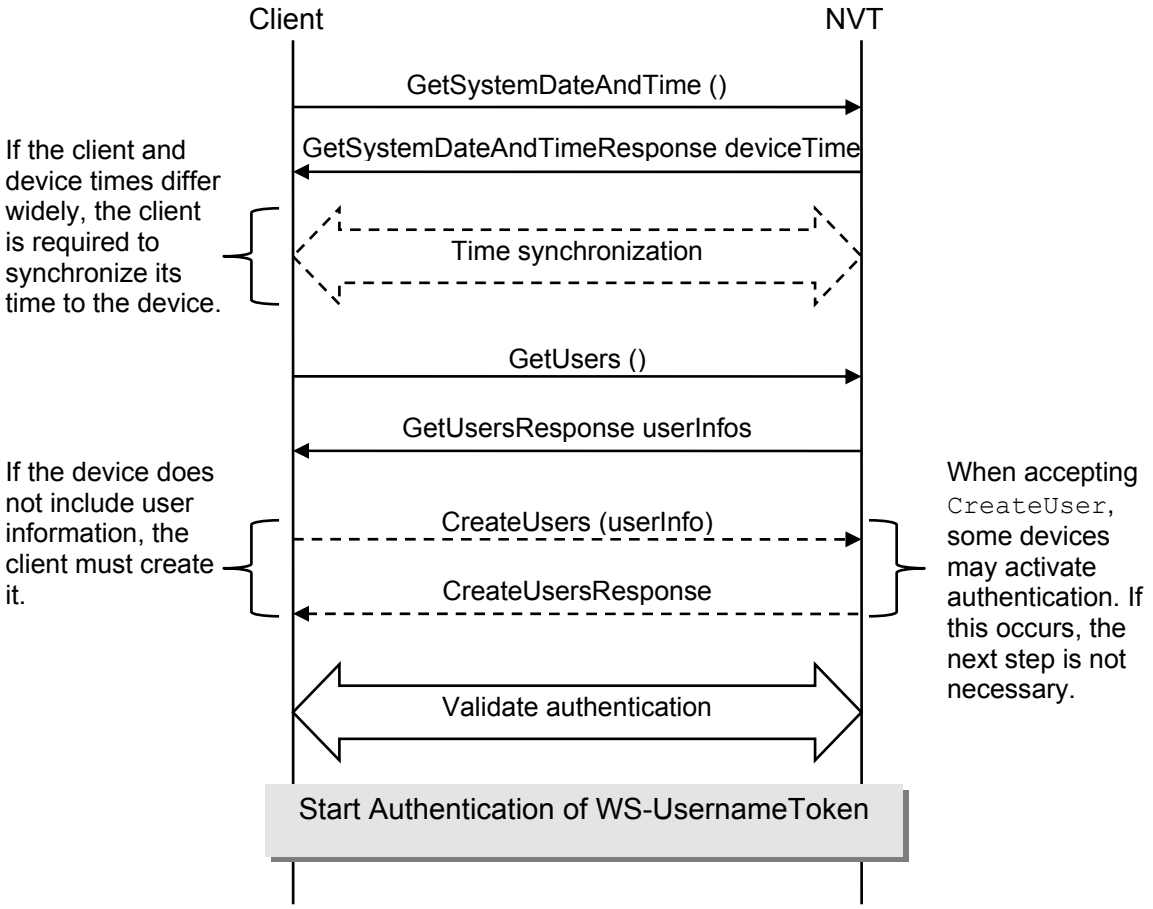
Response – on authentication error

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
        <env:Subcode>
          <env:Value>ter:NotAuthorized</env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en">Sender not Authorized</env:Text>
      </env:Reason>
      <env:Detail>
        <env:Text>The action requested requires authorization and the sender is not
authorized.</env:Text>
      </env:Detail>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

6.1.2 Validating WS-UsernameToken

When a device requries authentication based on `WS-UsernameToken`, the time setting of the client and of the device must be synchronized. Typically, the default state of a device will allow a certain set of anonymous access rights. A user must then authenticate to receive additional privileges on the device.

The [ONVIF\Core] specification states that a device without declared users should operate in a fully privileged mode. After users are created, then authentication policies should be applied. This example demonstrates how to ensure that a device requires authentication. It will create a user, if none exists, on the device to turn authentication on.



6.1.2.1 Prerequisites

- The device does not require authentication, for example, those with factory default settings where authentication has not been enabled.

6.1.2.2 Targeted Services and Technologies

- [ONVIF/Device management] and [devicemgmt.wsdl]

6.1.2.3 ONVIF::InitializeDeviceAuthentication

This use case explains what happens when the client validates the `WS-UsernameToken` authentication for the device.

Process:

1. The client obtains the time of the device through `GetSystemDateAndTime`.
2. The client compares its time setting to that of the device and synchronizes if necessary.

   If the difference between the time of the client and the device is too big, the device may refuse the `WS-UsernameToken`. In this case, the client can synchronize to match the device's time by using `SetSystemDateAndTime` or NTP.
3. The client checks to see if there are any users on the device. If there are none, then it establishes a new user to enable authentication on the device.

Privileged commands now require authentication using `WS-UsernameToken` as shown in Section 6.1, Authentication.

```
// create the device object to use the service
deviceService = getDeviceService(MyDeviceServiceAddress)

// The acquisition of the device time
deviceTime = deviceService.GetSystemDateAndTime();

// Comparison of the time
clientTime = getClientTime();
if(false == compareTime(deviceTime, clientTime))
{
//optionally synchronize the time of client and device by some kind of means
//See Chapter 5.4 and 5.5 on how to do
    somethingMethods.SynchronizeTime();
}

// The acquisition of the user info that is registered in a device
userInfos = deviceService.GetUsers();

// The confirmation that the client owns user info
hasUserInfo = false;
foreach(userInfo in userInfos)
{
    hasUserInfo = checkUserInfo(userInfo);
    if(true == hasUserInfo)
    {
        break;
    }
}

// User's registration(When it is necessary)
if(false == hasUserInfo)
{
    userInfo.Username = "newUsername";
    userInfo.Password = "newUserPassword";
    userInfo.UserLevel = "Administrator";
// When a device accept CreateUser, some device may activate authentication.
// In this case it is not necessary to execute next step.
deviceService.CreateUsers(userInfo);
}

// Validate WS-UsernameToken(the function is device dependence)
somethingMethods.ValidateWSUsernameToken();
```

6.2 User Management

This section describes setting up user information for authentication, emphasizing how to set the parameters of the commands for user registration.

6.2.1 Registering the User

Registering a user requires a device for user authentication.

6.2.1.1 Prerequisites

None.

6.2.1.2 Targeted Services and Technologies

- [ONVIF/Device management] and [devicemgmt.wsdl]

6.2.1.3 ONVIF::RegisteringUser

`CreateUsers` of `DeviceService` is used to register a user with a device. Three parameters in `CreateUsers` are set when users are registered:

- `Username` – Creates a user name for the new user
- `Password` – Creates a password for the user, entered in plain text
- `UserLevel` – Sets the authority level that controls the user's privileges

The client sets these parameters for each user and registers them by using `CreateUsers`.

```
// create the device object to use the service
deviceService = getDeviceService(MyDeviceServiceAddress)

// User's registration
registUserInfo.Username = "newusername";
registUserInfo.Password = "newuserpassword";
registUserInfo.UserLevel = "Administrator";

registUsers = {registUserInfo};

deviceService.CreateUsers(registUsers);
```

6.2.2 Changing the Password

6.2.2.1 Prerequisites

None.

6.2.2.2 Targeted Services and Technologies

- Device Service: see [ONVIF/Device management] and [devicemgmt.wsdl]

6.2.2.3 ONVIF::ChangingUserPassword

Use `SetUser` to change user information that is registered to the device. This example shows how to change a user's password.

Along with the user's registration, three parameters can be updated to change user information:

- `Username` – Changes the user name
- `Password` – Changes the password, entered in plain text
- `UserLevel` – Changes the authority level that controls privileges of the user

The client sets these parameters when user information is updated, and changes user information by using `SetUser`.

```
// create the device object to use the service
deviceService = getDeviceService(MyDeviceServiceAddress)

// A change of the user setting
changeUserInfo.Username = "username";
changeUserInfo.Password = "newpassword";
changeUserInfo.UserLevel = "Administrator";

changeUserInfos = {changeUserInfo};

deviceService.SetUser(changeUserInfos);
```

6.2.3 Deleting the User

6.2.3.1 Prerequisites

None.

6.2.3.2 Targeted Services and Technologies

- Device Service: see [ONVIF/Device management] and [devicemgmt.wsdl]

6.2.3.3 ONVIF::DeletingUser

`DeleteUsers` is used to delete a registered user from the device. The name of a user who will be deleted is set with the parameter `DeleteUsers`.

The client sets the user names to delete and deletes users in `DeleteUsers`.

CAUTION: Be aware that each device might behave differently when all administrators are deleted, or when the operator that executes `DeleteUsers` is deleted.

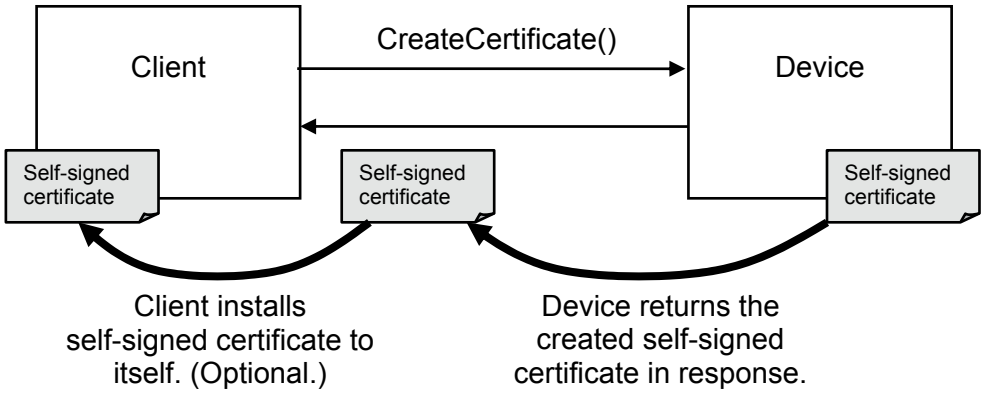
```
// create the device object to use the service
deviceService = getDeviceService(MyDeviceServiceAddress)

// The deletion of the user
deleteUserInfo.Username = "deleteusername";
deleteUserNameList = {deleteUserInfo.Username};

deviceService.DeleteUsers(deleteUserNameList);
```

6.3 Certificate Management and Usage

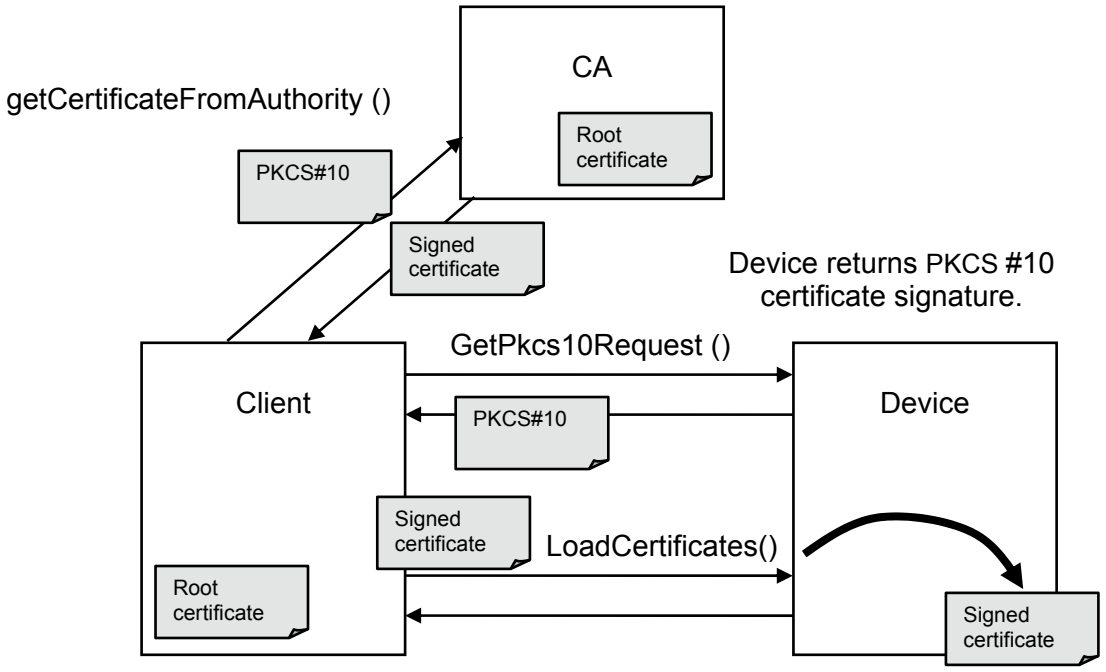
According to the ONVIF specification, a client can use TLS to connect to a device, and some methods are defined for setting up the TLS connection. The procedure for setting up a TLS connection differs depending on the type of authentication being used, but the specification does not define the details for how to use these methods to set up the connection. Therefore, this section provides an example of how to handle a certificate for server authentication of TLS.

Both “self-signed certificate” and “signed certificate in Certificate Authority (CA)” methods can be for TLS communication. The following figures show the basic architecture for using these certificates.

Using a self-signed certificate (see 6.3.1)



Using a signed certificate in CA (see 6.3.2 and 6.3.3)



6.3.1 Setting Up a Self-Signed Certificate of the Device

This section includes an example of a client setting a self-signed certificate for server authentication of TLS to a device.

6.3.1.1 Prerequisites

The device must support onboard key pair generation.

6.3.1.2 Targeted Services and Technologies

- Device Service (TLS): see [ONVIF/Device management] and [devicemgmt.wsdl]

6.3.1.3 ONVIF::SetSelfSignedCertificate

Although the following example describes how to set up a self-signed certificate, note that additional settings are required for a TLS connection, such as activation of HTTPS port by `SetNetworkProtocols`, and installation of the created Certificate onto the client.

```
// Create the device object to use the service
deviceService = getDeviceService(MyDeviceServiceAddress);

// The following procedures set the parameter of CreateCertificate request.

// Set the ID of the self-signed certificate of the device.
certificateID = "SelfSigned1";

// Set the subject parameter for CreateCertificateRequest.
// Settings for the value of Subject are vendor specific.
subject = App.SetSubject();

// ValidNotBefore and ValidNotAfter are optional parameters. Set these parameters,
// as required.
// ValidNotBefore and ValidNotAfter must be expressed in Greenwich Mean Time.
validNotAfter.Date.Year = 2020;
validNotAfter.Date.Month = 10;
validNotAfter.Date.Day = 1;
validNotAfter.Time.Hour = 9;
validNotAfter.Time.Minute = 0;
validNotAfter.Time.Second = 0;

// Create a self-signed certificate of the device as a result of onboard key pair
// generation.
// nvtCertificate : The result data of CreateCertificateResponse
// This message contains the generated self-signed certificate.
nvtCertificate =
deviceService.CreateCertificate(certificateID, subject, null, validNotAfter);

// The following procedures generate self-signed certificates.

// Get the status (enabled/disabled) of the device server certificates.
// certificateStatuses : Data from GetCertificatesStatus
// This message contains a list of the device server certificates referenced by
// ID and their status.
certificateStatuses = deviceService.GetCertificatesStatus();

// If the status of the generated self-signed certificate is disabled, change the
// status to enabled. Only one device server certificate can be enabled at a time.
// Therefore, if the status of the device server certificates is "enable," except for
// generated self-signed certificates, set the status to "disable".
// In the following procedures, note the following:
// 1. Confirm that the certificate which you intend to enable is valid within the
// period set by the parameters ValidNotBefore and/or ValidNotAfter.
```

```
// 2. Some devices may not be able to change CertificateStatuses for the certificate
// management policy. If necessary, confirm this policy with the manufacturer because
// deivce implementation can differ.

// selfSignedCertificateID : generated Certificate ID of self-signed certificate
// selfSignedCertificateID : the ID of self-signed certificate of the device
selfSignedCertificateID = nvtCertificate.CertificateID;

foreach (status in certificateStatuses) {
    if ( (status.CertificateID == selfSignedCertificateID) &&
         (status.Status == false) ) {
        status.Status = true;
    } else if ( (status.CertificateID != selfSignedCertificateID) &&
                (status.Status == true) ) {
        status.Status = false;
    }
}

// Set the status of the device server certificates.
deviceService.SetCertificatesStatus(certificateStatuses);
```

6.3.2 Getting a PKCS #10 Certificate Signature Request from the Device

This section includes an example of a client obtaining a PKCS #10 certificate signature for server authentication of TLS from a device.

6.3.2.1 Prerequisites

- The device must support onboard key pair generation.
- A self-signed certificate is installed (see Section 6.3.1, Setting Up a Self-Signed Certificate of the Device).

6.3.2.2 Targeted Services and Technologies

- Device Service (TLS): see [ONVIF/Device management] and [devicemgmt.wsdl]

6.3.2.3 ONVIF::GetSignedCertificateRequest

To send a `GetPkcs10Request`, the parameter of this command requires setting an appropriate value. The following is an example of how to set this parameter.

Note that the Subject field in this example is vendor-specific and has been omitted. For support on how to create the subject for the specific request, follow up with the appropriate vendor.

```
// Create the device object to use the service
deviceService = getDeviceService(MyDeviceServiceAddress);

// The following procedures, set the parameter of GetPkcs10Request request.

// Set the ID of self-signed certificate of the device as a result of execution of
// the use case "Setup self-signed certificate of the device".
certificateID = "SelfSigned1";

// Set the subject parameter for GetPkcs10RequestReqeust.
// How to set the value of subject is vendor specific.
// Confirm required X.500 attribute type in the CA which issues signed certificate
// of the device

// Confirm required Attributes in the CA which issues signed certificate of the
device
// These attributes needs to be encoded as DER ASN.1 objects.
// DER() : The function to encode DER ASN.1
requiredAttributes = "Required Attributes";
attributes = DER(requiredAttributes);

// Request a PKCS #10 certificate signature request from the device.
// nvtCertificate : The message of GetPkcs10RequestResponse
// This message contains the PKCS#10 request data structure.
pkcs10Request = deviceService.GetPkcs10Request(certificateID, subject, attributes);
```

6.3.3 Setting Up a Signed Certificate of the Device (Except for a Self-Signed Certificate)

This section includes an example of a client setting up the certificate that was signed by CA for server authentication of TLS to a device.

6.3.3.1 Prerequisites

- A self-signed certificate is installed (see Section 6.3.1, Setting Up a Self-Signed Certificate of the Device).
- The signed certificate of the device using the PKCS#10 certificate request has been received from CA

6.3.3.2 Targeted Services and Technologies

- Device Service (TLS) see [ONVIF/Device management] and [devicemgmt.wsdl]

6.3.3.3 ONVIF::SetCASingedCertificate

Although the following example describes how to set up the certificate that was signed by CA, note that additional settings are required for a TLS connection.

```
// Create the device object to use the service
deviceService = getDeviceService(MyDeviceServiceAddress);

// The following procedures, set the parameter of LoadCertificates request.

// Set the ID of signed certificate of the device.
nVTCertificate.CertificateID = "CASigned1";

// Get the signed certificate from CA. This operation may execute by off line.
nVTCertificate.CACertificate = App.getCertificateFromAuthority();

// Load signed certificate of the device into the device.
deviceService.LoadCertificates(nVTCertificate);

// The following procedures make the signed certificate enabled.

// get the status (enabled/disabled) of the device server certificates.
// getCertificateStatuses : The parameter of CreateCertificateResponse
// This message contains a list of the device server certificates referenced by ID
// and their status.
certificateStatuses = deviceService.GetCertificatesStatus();

// If the status of the signed certificate is disabled, change the status to enabled.
// Only one device server certificate can be enabled at a time. Therefore,
// if the status of device server certificates is "enable", except for signed
// certificates, set the status to "disable".
// In the following procedures, note the following:
// 1. Confirm that the certificate which you intend to enable is valid within the
// period which is set by the parameters ValidNotBefore and/or ValidNotAfter.
// 2. Some devices may not be able to change CertificateStatuses for the certificate
// management policy. If necessary, confirm this policy with the manufacturer because
// device implementation can differ.


foreach (status in certificateStatuses) {
    if ( (status.CertificateID == nVTCertificates.CertificateID) &&
         (status.Status == false) ) {
        status.Status = true;
    } else if ( (status.CertificateID != nVTCertificates.CertificateID) &&
```

```
                (status.Status == true) ) {
        status.Status = false;
    }
}

// Set the status of the device server certificates.
deviceService.SetCertificatesStatus(certificateStatuses);
```

6.3.4 Getting Information About Device Certificates

This section includes an example of a client getting the information about a specific, installed certificate from a device.

6.3.4.1 Targeted Services and Technologies

- Device Service (TLS): see [ONVIF/Device management] and [devicemgmt.wsdl]

6.3.4.2 ONVIF::GetCertificateInformation

To get information on a specific certificate, you must obtain the `certificateID`. This can be easily obtained using the `getCertificateStatuses` function in this example.

```
// Create the device object to use the service
deviceService = getDeviceService(MyDeviceServiceAddress);

// get the status (enabled/disabled) of the device server certificates.
// getCertificateStatuses : The parameter of CertificateStatusesResponse
// This message contains a list of the device server certificates referenced by
// ID and their status.
certificateStatuses = deviceService.GetCertificatesStatus();

// Check certificate, the information of the first certificates is gotten
// if a certificate exists
if(present(certificateStatuses))
{
    // certificateInformation : The result data of GetCertificateInformationResponse
    certificateInformation =
      deviceService.GetCertificateInformation(certificateStatuses[0].CertificateID);
}
```

6.3.5 Deleting the Certificates of a Device

This section includes an example of a client deleting a certificate that is no longer needed (invalid or for other reasons) from a device.

6.3.5.1 Prerequisites

- The device contains at least one server certificate in addition to the one which will be deleted.
- The client has the `CertificateIDs` of the certificates which he wants to delete and to activate. How to get these is described in the previous use case (see Section 6.3.4, Getting Information About Device Certificates).

6.3.5.2 Targeted Services and Technologies

- Device Service (TLS): see [ONVIF/Device management] and [devicemgmt.wsdl]

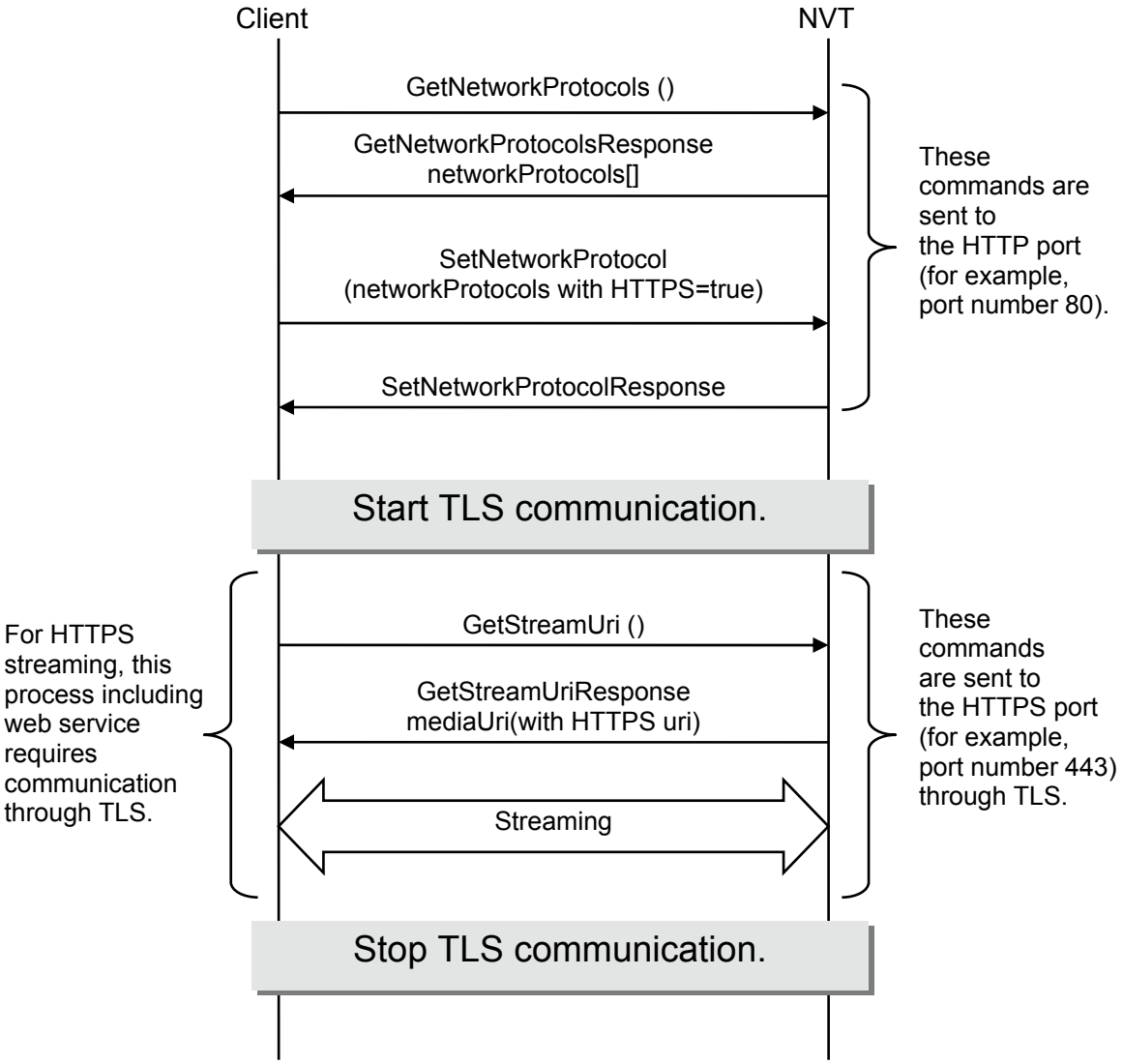
6.3.5.3 ONVIF::DeleteCertificate

```
// Create the device object to use the service
deviceService = getDeviceService(MyDeviceServiceAddress);

// DeleteCertificateID: the ID of the certificate which is intended to delete.
deleteCertificateID = "DeleteID";
// EnableCertificateID: the ID of the certificate which is intended to enable.
enableCertificateID = "EnableID";


// Get the status (enabled/disabled) of the device server certificates.
// getCertificateStatuses : The result data of CreateCertificateResponse
// This data contains a list of the device server certificates referenced by ID
// and their status.
certificateStatuses = deviceService.GetCertificatesStatus();

// If the status of the certificate whose ID is enableCertificateID is enable, set
// the status disable.
// Only one device server certificate is allowed to be enabled at a time. Therefore,
// if the status of device server certificates except the certificate whose ID is
// enableCertificateID is enable, set the status disable.
// The following procedures, be sure to note the following points.
// 1. Confirm that the certificate which is intended to enable is within the validity
// period which is set to the parameters ValidNotBefore and/or ValidNotAfter.
// 2. Some devices may not be able to change CertificateStatuses for the certificate
// management policy. Please confirm to the maker this policy if necessary since it
is
// different by implementation of a device.

foreach (status in certificateStatuses) {
    if ( (status.CertificateID == enableCertificateID) &&
         (status.Status == false) ) {
        status.Status = true;
    } else if ( (status.CertificateID != enableCertificateID) &&
                (status.Status == true) ) {
        status.Status = false;
    }
}

// Set the status of the device server certificates.
deviceService.SetCertificatesStatus(certificateStatuses);

// Delete the certificate
// Some devices may not be able to delete CertificateStatuses for the certificate
```

```
// management policy. Please confirm to the maker this policy if necessary since it
is
// different by implementation of a device.

deviceService.DeleteCertificates(deleteCertificateID);
```

6.4 Real-Time Streaming via RTP / RTSP / HTTPS

According to the ONVIF specification, clients can get real-time streaming via TLS. However, the specification does not provide a parameter for specifying HTTPS in `tt:TransportProtocol`, so implementing this feature is not obvious. This section provides an example of how to get Real-time streaming via RTP/RTSP/HTTPS (the procedure for HTTP and RTSP authentication, however, is not described here).



6.4.1 Prerequisites

- The signed or self-signed certificate of TLS must already be created and enabled via the `SetCertificateStatus` command. That is, the certificate should be ready to turn on HTTPS.
- Confirm that the device and the client support TLS versions 1.0, 1.1, or 1.2.

6.4.2 Targeted Services and Technologies

- Device Service: see [ONVIF/Device management] and [devicemgmt.wsdl]
- Media Service: see [ONVIF/Media] and [media.wsdl]

6.4.3 ONVIF::RealTimeStreaming_RTP_RTSP_HTTPS

When the client receives streaming over HTTPS, TLS and the controlling stream needs to be set.

Process:

1. Set up an HTTPS connection.

   If the network setting of HTTPS is not enabled, this setting is required.

2. Initialize the real-time stream of RTP/RTSP/HTTPS.

```
// create the device object to use the Service
deviceService = getDeviceService(MyDeviceAddress);

// The client gets the status (Enabled) of HTTPS and the port number by
// GetNetworkProtocols.
networkProtocols = deviceService.GetNetworkProtocols();

// The client sets the status(Enabled) of HTTPS to "true" due to activate HTTPS.
foreach (networkProtocol in networkProtocols)
{
    if(networkProtocol.Name == "HTTPS"){
        networkProtocol.Enabled = true;
        networkProtocol.Port[0] = 443;
    }
}

deviceService.SetNetworkProtocols(networkProtocols);

// The client will wait a few minutes before doing next step
// because the device may reboot. (This behaviour depends on the device).
somethingMethod.wait();

// Set the start of TLS connection for the client.
// The following all http connection including web service will become through TLS.
somethingMethod.startTLS();

// create the media object to use the service
mediaService = getMediaService(MyMediaServiceAddress);

// The client gets and chooses the profile.
ProfileList = mediaService.GetProfiles();

// use the first profile (the device has at least one media profile).
targetProfileToken = ProfileList[0].token;

// set the GetStreamUriRequest paramater
streamSetup.Stream = "RTP-Unicast";
streamSetup.Transport.Protocol = HTTP;
streamSetup.Transport.Tunnel = null;

// Get the HTTPS URI by GetStreamUri.
// The client will receive the URI address that starts by "https"
// (i.e. URI = https://*********).
mediaUri = mediaService.GetStreamUri(streamSetup, targetProfileToken);
```

```
// The client sets the URI address for HTTPS stream.
App.DoStreaming(mediaUri.Uri);

// end the streaming
App.StopStreaming(mediaUri.Uri);

// end of TLS connection
somthingMethod.stopTLS();
```

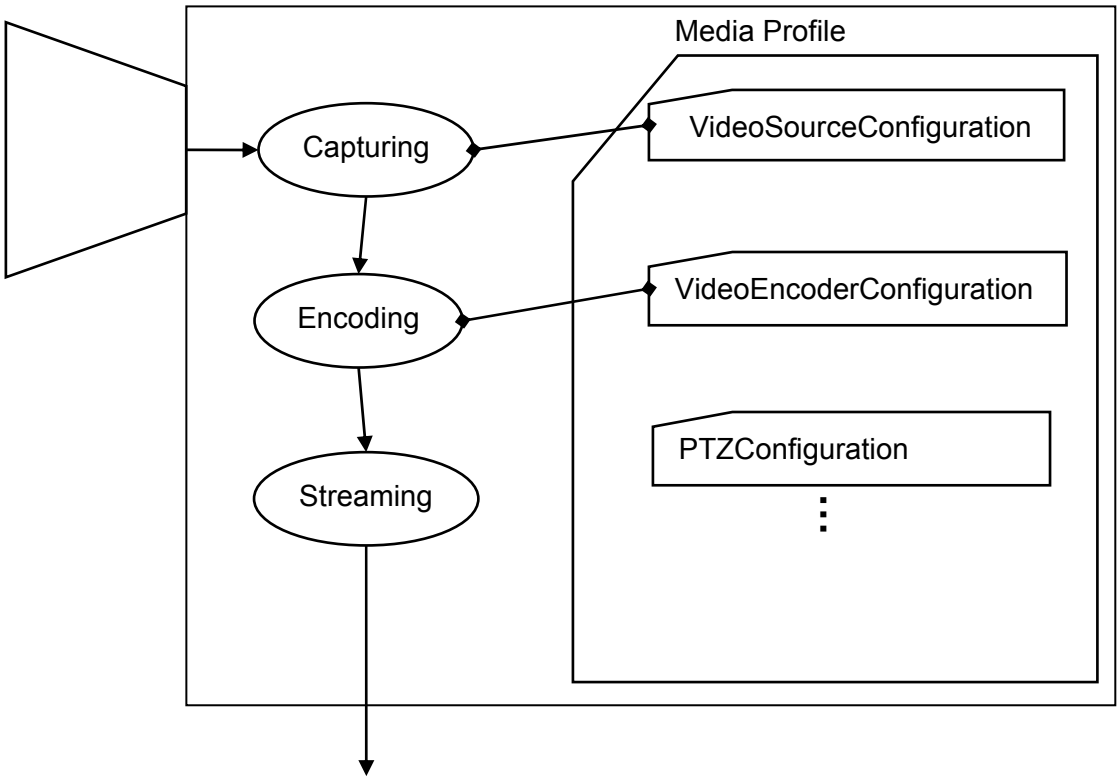
7 Streaming

This chapter describes the real-time video and audio streaming function in the ONVIF specification. These configurations are controlled using media profiles. Each configuration becomes effective when it is added to a profile. The following diagram shows a media profile.

A media profile includes the following for configuring video streaming capabilities:

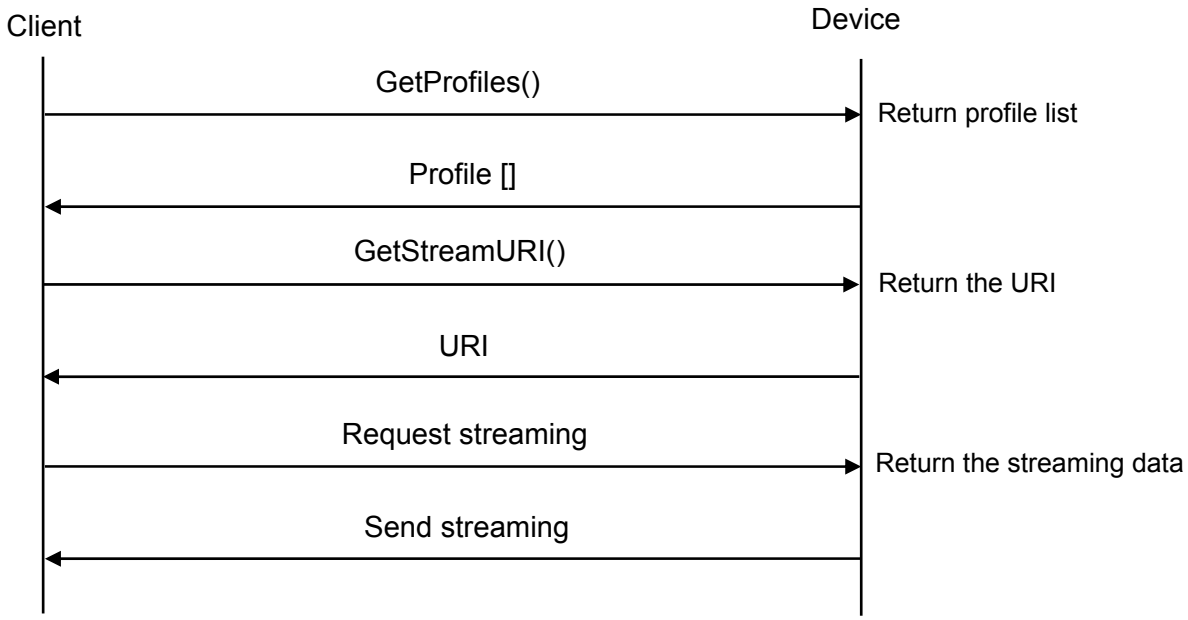
- `VideoSourceConfiguration` – Contains a reference to a `VideoSource` and a `Bounds` structure containing either the entire VideoSource pixel area or a sub-portion.
- `VideoEncoderConfiguration` – Contains encoding data that consists of codec, pixel resolution, and quality specifications.

Receiving media streaming involves getting a stream URI from a certain media profile.


Media Profile
Capturing
Encoding
Streaming
VideoSourceConfiguration
VideoEncoderConfiguration
PTZConfiguration

7.1 Using an Existing Profile for Media Streaming

A device with a media configuration service enabled includes at least one media profile at boot. This use case shows how to start video streaming with UDP Unicast by using an existing media profile.



7.1.1 Prerequisites

At least one media profile is available that contains `VideoSourceConfiguration` and `VideoEncoderConfiguration`.

7.1.2 Targeted Services and Technologies

- [ONVIF/Media] and [media.wsdl]

7.1.3 ONVIF::StartUdpUnicastStream

This example involves:

1. Creating a media object to use the service. This is done from the WSDL by the underlying framework, and results in a `MediaService` object that encapsulates the service functions as methods.

2. Asking the `MediaService` for the existing profiles. The first profile gets its `ReferenceToken` for the stream setup.

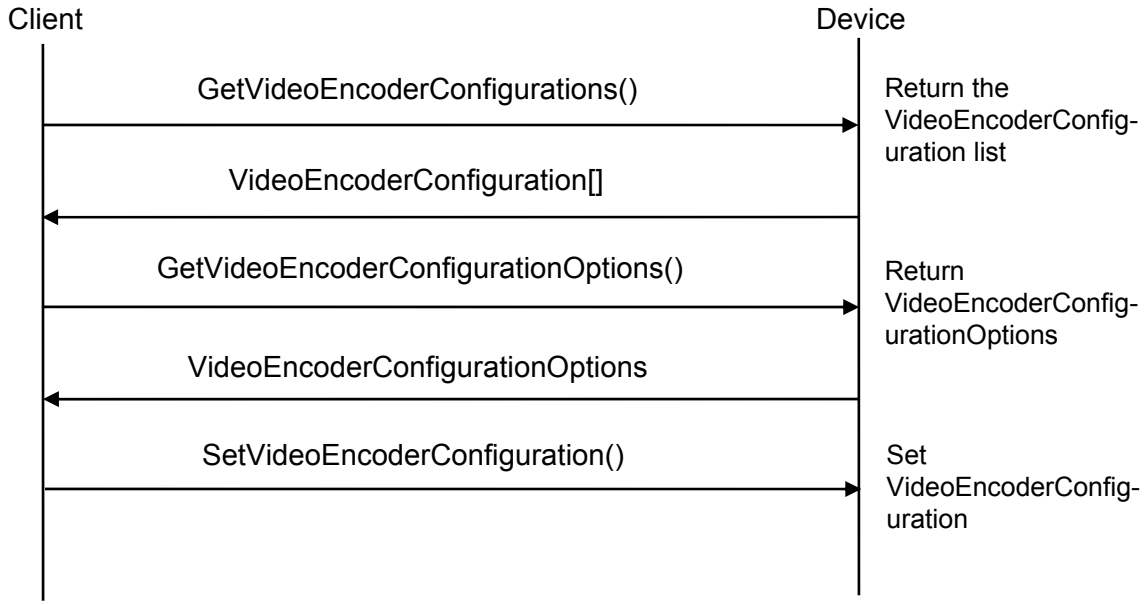
3. Creating a `StreamSetup` with, for example, `RTP-Unicast` as the stream type, `UDP` as transport, and no tunnelling.

4. Using the `ProfileToken` and the `StreamSetup` to retrieve the `StreamUri` which can be used to obtain the video stream.

```
// create the media object to use the service
mediaService = getMediaService(MyMediaServiceAddress);

// get profiles
profilesList = mediaService.GetProfiles();

// use the first profile and Profiles have at least one
mediaProfileToken = profilesList[0].token;

// setup stream configuration
streamSetup.Stream = "RTP-Unicast";
streamSetup.Transport.Protocol = "UDP";

// RTP/RTSP/UDP is not a special tunnelling setup (is not requiring)!
streamSetup.Transport.Tunnel = null;

// get stream URI
mediaUri = mediaService.GetStreamUri(streamSetup, mediaProfileToken);

App.DoStreaming(mediaUri.Uri);
```

7.2 Media Profile Configuration

A media profile consists of configuration entities such as video/audio source configuration, video/audio encoder configuration, or PTZ configuration. This use case describes how to change one configuration entity which has been already added to the media profile.



7.2.1 Prerequisites

At least one `VideoEncoderConfiguration` is available in the media service.

7.2.2 Targeted Services and Technologies

- [ONVIF/Media] and [media.wsdl]

7.2.3 ONVIF::ChangeMediaProfile

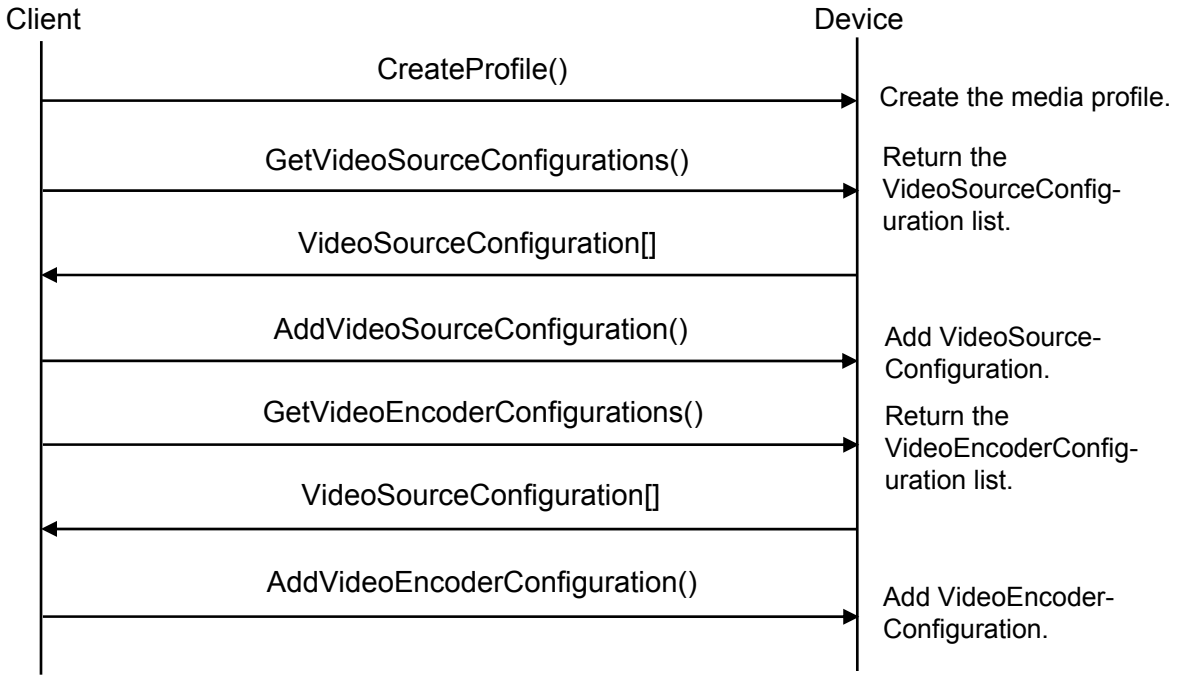
This example describes changing video `codec` to `jpeg`:

1. Creating a media service and copying `VideoEncoderConfiguration` which was previously added to it.
2. Getting video encoder capability by using `GetVideoEncoderConfigurationOptions`, which sets a range for each codec.
3. Changing `Encoding`, `Resolution`, `Quality`, and `RateControl` within the range in `VideoEncoderConfigurationOptions`.
4. Using `videoEncoderConfiguration` and `forcePersistence` to retrieve `SetVideoEncoderConfiguration`.

```
// create the media object to use the service
mediaService = getMediaService(MyMediaServiceAddress);

// get all video encoder configurations
configurationsList = mediaService.GetVideoEncoderConfigurations();

// use the first configuration and VideoEncoderConfigurations have at least one
videoEncoderConfiguration = configurationsList[0];

// get video encoder configuration options
Options = mediaService.GetVideoEncoderConfigurationOptions(mediaConfiguration.token);


// set video encoder configuration using video encoder configuration options
videoEncoderConfiguration.Encoding = "JPEG";
videoEncoderConfiguration.Resolution.Width =
Options.JPEG.ResolutionsAvailable[0].Width;
videoEncoderConfiguration.Resolution.Height =
Options.JPEG.ResolutionsAvailable[0].Height;
videoEncoderConfiguration.Quality = Options.QualityRange.Min;
videoEncoderConfiguration.RateControl.FrameRateLimit =
Options.JPEG.FrameRateRange.Min;
videoEncoderConfiguration.RateControl.EncodingInterval =
Options.JPEG.EncodingIntervalRange.Min;
videoEncoderConfiguration.RateControl.BitrateLimit =
Options.Extention.Jpeg.BitrateRange.Min;

forcePersistence = true;

// set the video encoder configuration
mediaService.SetVideoEncoderConfiguration(videoEncoderConfiguration,
forcePersistence);
```

7.3 Creating a New Media Profile and Adding an Entity

The NVT presents different available profiles depending on its capabilities. This use case describes how to create a new media profile. It is useful when, for example, we receive multiple streaming.



7.3.1 Prerequisites

- At least one `VideoSourceConfiguration` and `VideoEncoderConfiguration` must be available in the media service.

7.3.2 Targeted Services and Technologies

- [ONVIF/Media] and [media.wsdl]

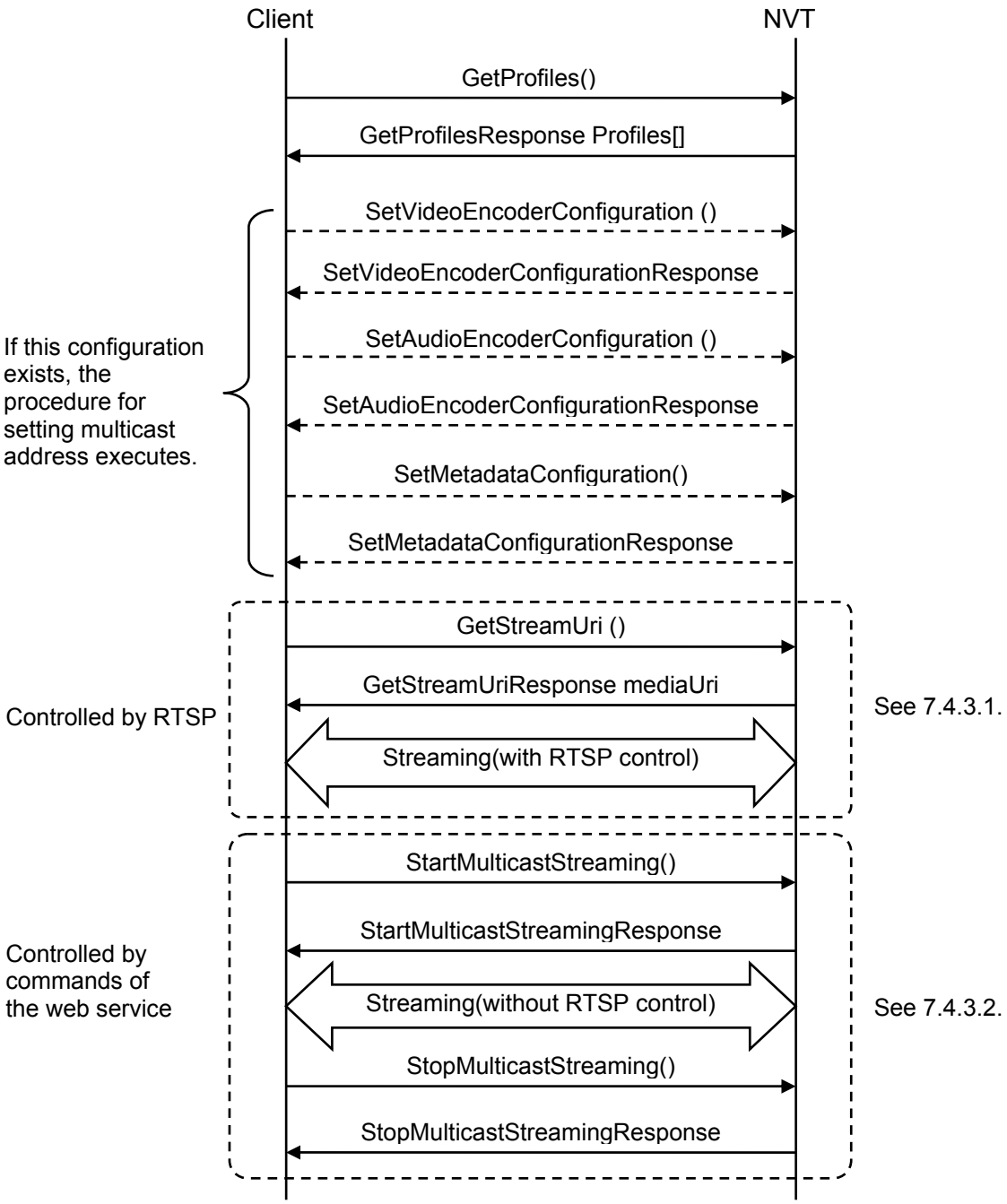
7.3.3 ONVIF::CreateMediaProfile

This example describes how to create a new media profile, and how to set the H.264 `codec` which has been already created in `VideoEncoderConfguration`. First, a new media profile is created. A video source configuration using the new media profile uses one that already exists. A video encoder configuration uses one that already exists.

```
// create the media object to use the service
mediaService = getMediaService(MyMediaServiceAddress);

// create a new profile token
name = "profileName";
token = "profileToken";
mediaProfile = mediaService.CreateProfile(name, token)

// video source configuration must be added first.
// get all video source configurations
sourceConfigurationsList = mediaService.GetVideoSourceConfigurations();

//use the first configuration and SourceEncoderConfigurations have at least one
sourceConfigurationToken = sourceConfigurationsList[0].token;
mediaService.AddVideoSourceConfiguration(ProfileToken, sourceConfigurationToken);

// add video encoder configuration later
// get all video encoder configurations
encoderConfigurationsList = mediaService.GetVideoEncoderConfigurations();

//search H.264 streaming
foreach( encoderConfiguraton in encoderConfigurationsList)
{
  if(encoderConfiguraton.Encoding == "H264")
  {
    // add video encoder configuration
    encoderConfigurationToken = encoderConfigurationsList.token;

    mediaService.AddVideoEncoderConfiguration(ProfileToken,encoderConfigurationToken)
    ;
    break;
  }
}
//now the stream can be started as already shown in chapter 7.1
```

7.4 Multicast Streaming

According to the ONVIF specification, a client can control multicast streaming of a device, and some methods are defined for multicast streaming setup and control. A client needs to specify how to control multicast streaming. This section provides two samples for IPv4 streaming where a client sets multicast streaming configuration and controls the RTP stream. Also, a “bad practice” for the multicast stream setting is described (but not recommended).



7.4.1 Prerequisites

- The device must support multicast streaming.
- The `GetCapabilities` response from the device contains: `Media - StreamingCapabilities - RTPMulticast = true`

7.4.2 Targeted Services and Technologies

- Media Service: see [ONVIF/Media configuration] and [media.wsdl]
- Real-time streaming: see [ONVIF/Realtime-Streaming]

7.4.3 ONVIF::MulticastStreaming

In this section, all configurations relating to one media profile are set for multicast streaming. See sections 7.4.3.1, Case of with RTSP, and 7.4.3.2, Case of StartMulticastStreaming Command, for descriptions of multicast control.

In this example:

1. The client sends a `GetProfilesRequest` to select a profile.
2. The client confirms that the `Multicast` parameter is included in all the configurations that are added to the selected profile: `VideoEncoderConfiguration`, `AudioEncoderConfiguration`, `MetadataConfiguration`
3. If the multicast address, port, and TTL are not set (`IPv4Address = 0.0.0.0`, `Port = 0`, `TTL = 0`), the client sets these parameters using: `Set<configuration entity>Configuration`

These parameters should be set in all of the configurations that are added to the profile.

```
// create the media object to use the service
mediaService = getMediaService(MyMediaServiceAddress);

// get profiles
profilesList = mediaService.GetProfiles();

// use the first profile and Proiles have at least one
mediaProfileToken = profilesList[0].token;

// The client confirms Multicast parameter inside all of configurations that are
added
// to the selected profile.

// check of VideoEncoderConfiguration
if( present(profilesList[0].VideoEncoderConfiguration) )
{
    videoEncoderConfiguration = profilesList[0].VideoEncoderConfiguration;
    multicastConfiguration = videoEncoderConfiguration.Multicast;

    // set these parameters for UDP multicast.
    multicastConfiguration.Address.Type = "IPv4";
    multicastConfiguration.Address.IPv4Address = "239.192.10.10";
    multicastConfiguration.Port = 30000;
    multicastConfiguration.TTL = 5;
    videoEncoderConfiguration.Multicast = multicastConfiguration;
    mediaService.SetVideoEncoderConfiguration(videoEncoderConfiguration, true);
```

```
}

// check of AudioEncoderConfiguration
if( present(profilesList[0].AudioEncoderConfiguration) )
{
    audioEncoderConfiguration = profilesList[0].AudioEncoderConfiguration;
    multicastConfiguration = audioEncoderConfiguration.Multicast;

    // set these parameters for UDP multicast.
    multicastConfiguration.Address.Type = "IPv4";
    multicastConfiguration.Address.IPv4Address = "239.192.10.10";
    multicastConfiguration.Port = 30002;
    multicastConfiguration.TTL = 5;
    videoEncoderConfiguration.Multicast = multicastConfiguration;
    mediaService.SetAudioEncoderConfiguration(videoEncoderConfiguration, true);
}

// check of MetadataConfiguration
if( present(profilesList[0].MetadataConfiguration) )
{
    metadataConfiguration = profilesList[0].MetadataConfiguration;
    multicastConfiguration = metadataConfiguration.Multicast;

    // set these parameters for UDP multicast.
    multicastConfiguration.Address.Type = "IPv4";
    multicastConfiguration.Address.IPv4Address = "239.192.10.10";
    multicastConfiguration.Port = 30004;
    multicastConfiguration.TTL = 5;
    videoEncoderConfiguration.Multicast = multicastConfiguration;
    mediaService.SetMetadataConfiguration(videoEncoderConfiguration, true);
}
```

TIP: If you want to disable a multicast configuration, the address field of <*configuration entity*>Configuration should be set to 0.0.0.0.

7.4.3.1 Case of with RTSP

This chapter describes the method of playing by RTSP control. When the previous procedure is complete, the process continues in the following procedure.

```
// Setup stream configuration
streamSetup.Stream = "RTP-Multicast";
streamSetup.Transport.Protocol = "UDP";
streamSetup.Transport.Tunnel = null;

// Get stream URI
mediaUri = mediaService.GetStreamUri( streamSetup, mediaProfileToken );

app.doStreaming(mediaUri.Uri, "Multicast");
```

7.4.3.2 Case of StartMulticastStreaming Command

This chapter describes the method of playing by `StartMulticastStreaming`. When the previous procedure is complete, the process continues in the following procedure.

```
// Start receiving multicast
if( present(profilesList[0].VideoEncoderConfiguration) )
{
    appVideo.receiveStreaming("239.192.10.10:30000");
}
if( present(profilesList[0].AudioEncoderConfiguration) )
{
    appAudio.receiveStreaming("239.192.10.10:30002");
}
if( present(profilesList[0].MetadataConfiguration) )
{
    appMetadata.receiveStreaming("239.192.10.10:30004");
}

// Start mulitcast streaming
mediaService.StartMulticastStreaming( mediaProfileToken );

// Wait for stop request
app.waitForStopRequest();

// Stop mulitcast streaming
mediaService.StopMulticastStreaming( mediaProfileToken );
```

NOTICE: If all the configurations that are added to the profile do not contain the multicast setting, the device might respond with the fault code: `env:Receiver/ter:Action/ter:IncompleteConfiguration`

NOTICE: If the client requests the `Add<`*`configuration entity`*`>Configuration` or `Remove<`*`configuration entity`*`>Configuration` command with the profile which already started multicast streaming, the device might stop multicast streaming.

7.4.4 RSTP Communication Trace

[ONVIF/Example: Multicast Setup]

7.4.5 Bad Practice of Multicast Streaming

In some cases, the client may not be able to play due to a mistake in a configuration setting. This section describes an example of this “bad practice.” The following procedure is *not recommended*.

Assigning a Configuration to Two or More Profiles

If the same `VideoEncoderConfiguration#1` is added to both `Profile1` and `Profile2`, the device might not be able to start multicast streaming. In this case, both `Profile1` and `Profile2` are assigned the same multicast address (for example, `239.192.10.10:30000`).

7.5 Audio Backchannel Handling

This use case shows how a bidirectional audio connection could be established using the ONVIF RTSP extension. The NVT in this example provides one audio output that can be connected to a loudspeaker. It may be able to decode G.711, G.726, or AAC audio. The client is able to stream G.711 audio.

1. The necessary settings for stream setup are set up. The client uses the device I/O service to request the available audio outputs and their configurations.

2. An existing media profile that is already configured with `VideoSourceConfiguration` and `VideoEncoderConfiguration`, as well as `AudioSource`- and `AudioEncoderConfiguration`, is used. To configure this profile for a backchannel connection, a suitable `AudioDecoder` and `AudioOutputConfiguration` is added.

   No parameters are available to configure the decoder. The client asks for the decoding capabilities of the specific configuration, and selects a suitable one that supports G.711 encoding.

3. After requesting the stream URI, an RTSP connection is established.

   The ONVIF core specification already includes an example of how a Unicast RTSP connection with backchannel support is established. To provide additional information, this use case establishes an HTTP tunnelled RTP connection.

7.5.1 Prerequisites

- An existing media profile that is already configured with `VideoSourceConfiguration` and `VideoEncoderConfiguration`, as well as `AudioSourceConfiguration` and `AudioEncoderConfiguration`

7.5.2 Targeted Services and Technologies

- [CORE/Media] and [media.wsdl]
- [Core/DeviceIO] and [deviceio.wsdl]
- [RTSP]

7.5.3 ONVIF::StartBackChannelStream

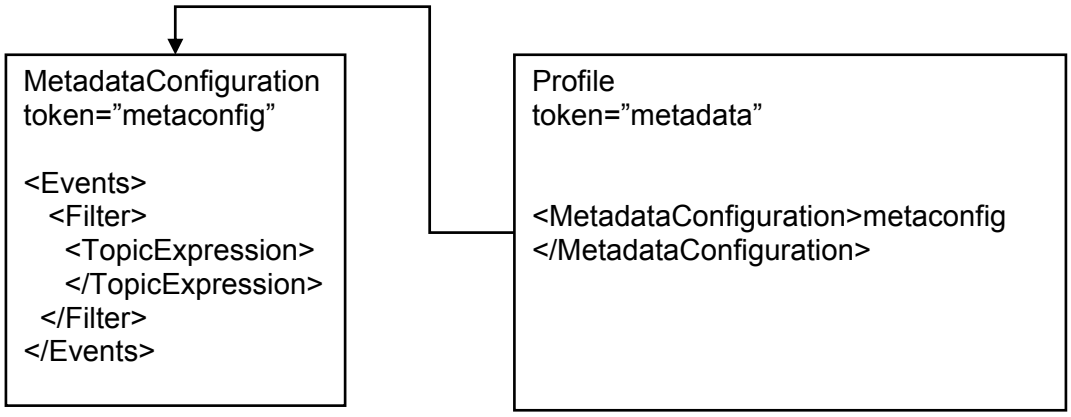
In the following example:

1. This function requests the existing `AudioOutputs` of the device.
2. The function asks for `AudioOutputConfigurations`, `AudioDecoderConfigurations`, and the decoding capabilities.
3. It requests the Stream uri for the media profile and starts the connection with the bi-directional audio stream.

```
// create the needed media object and DeviceIO object to use these services using the
// MediaServiceAddress and the DeviceIOAddress that can be requested using the
// GetCapabilities command
mediaService    = getMediaService(MyMediaServiceAddress);
deviceIOService = getDeviceIOService(MyDeviceIOServiceAddress);

//The device has one audio output. Request this audio output
audioOutList = deviceIOService.GetAudioOutputs();
audioOutput = audioOutList[0];

//Search an AudioOutputConfiguration (aoc) which is applicable for the AudioOutput
audioOutConfiguration =
deviceIOService.GetAudioOutputConfiguration(audioOutput.token);

// Get a G711 AudioEncoderConfiguration (adc) that supports G.711 encoding
audioDecoderConfigurationList = mediaService.GetAudioDecoderConfigurations();

foreach( adc in audioDecoderConfigurationList )
{
    adco = mediaService.GetAudioDecoderConfigurationOptions(adc.token);

    if( present( adco.G711DecOptions ) )
    {
        audioDecoderConfiguration = adc;
        break;
    }
}

//Add the selected AudioOutputConfiguration and AudioDecoderConfiguration
//to the Profile with token "Profile1"
mediaService.AddAudioOutputConfiguration("Profile1", audioOutputConfiguration.token);
mediaService.AddAudioDecoderConfiguration("Profile1",
audioDecoderConfiguration.token);

//ask for stream uri to setup the RTSP connection
streamSetup.Stream            = MediaService.StreamType.RTPUnicast;
streamSetup.Transport.Protocol = MediaService.TransportProtocol.HTTP;

//RTP/RTSP/HTTP is not a special tunnelling setup (is not requiring)!
streamSetup.Transport.Tunnel   = null;

uri = mediaService.GetStreamUri(streamSetup,"Profile1");

App.doStreaming(uri);
```

7.6 Setting Up Metadata Streaming

Metadata streaming is a way to receive event notifications in real-time over an RTP or RTSP stream. The transport can be included in a number of protocols supported by the device: RTP, RTP multicast, RTP over RTSP, and RTP over RTSP over HTTP. First, a media profile is set up that contains a `MetadataConfiguration` with the desired event filter. After that, the stream URI for that profile can be fetched and used. For more information regarding event notification, see Chapter 9, Eventing.



7.6.1 Prerequisites

- A device must already be discovered and the service URIs must be known.

7.6.2 Targeted Services and Technologies

- [ONVIF/Event-Handling] and [event.wsdl]
- [ONVIF/Media] and [media.wsdl]

7.6.3 ONVIF::TestMetadataStreaming

This example sets up a `MetadataConfiguration` and a `MediaProfile`, then calls `GetStreamUri` to get the URI to use.

The process involves getting the list of profiles using `media.GetProfiles`, then getting the `MetadataConfigurations` list using `media.GetMetadataConfigurations`.

```
// Tests ONVIF metadata streaming
test_metadata_streaming(onvifdev)
 {
   media = getMediaService()(onvifdev.mediaxaddr);
   ProfilesList = media.GetProfiles();
   aMetadataConfigurationsList = media.GetMetadataConfigurations();
   // Get the token for the first configuration
   // and then configure for our needs
   aMetadataConfiguration = aMetadataConfigurationsList[0];
   metadatatoken = aMetadataConfiguration.token; // Attribute



   // See if our desired metadataconfiguration already exists
```

```
aConfiguration =
  media.GetMetadataConfiguration(ConfigurationToken = metadatatoken);
aConfiguration.Name = "metadata";
// Must setup an Events Filter to get any events
// Set Dialect attribute
aConfiguration.Events.Filter.TopicExpression.Dialect =
  "http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet";
// SetTopicExpression to get everything under Device topic
aConfiguration.Events.Filter.TopicExpression = "tns1:Device//.";
// An alternative way to get all events would be to use:
// aConfiguration.Events.Filter.TopicExpression = null;

media.SetMetadataConfiguration(Configuration = aConfiguration);

// See if our desired profile already exists
Profile = media.GetProfile(ProfileToken = "metadata");
if (!Profile) {
  // No, our desired Profile does not exist,
  // use CreateProfile to create it:
  // CreateProfile with Name and Token:
  Profile = media.CreateProfile(Name = "metadata apg", Token = "metadata")
  // We get an empty Profile back
}

// Check if we have a MetadataConfiguration in the Profile:
// Not a strict requirement since we could choose to always add it,
// but lets save that call if we can.
if (!Profile.MetadataConfiguration) {
  // No MetadataConfiguration, lets add it:
  // AddMetadataConfiguration with our Profile(Token) and
  // Metadata ConfigurationToken
  media.AddMetadataConfiguration(ProfileToken = "metadata",
                                 ConfigurationToken = metadatatoken);
}
```

7.6.4 ONVIF::FetchMetadataStream

When a profile with `Metadataconfiguration` is set up, the stream can be set up like a typical video stream except that the payload in the stream will be XML with Notifications.

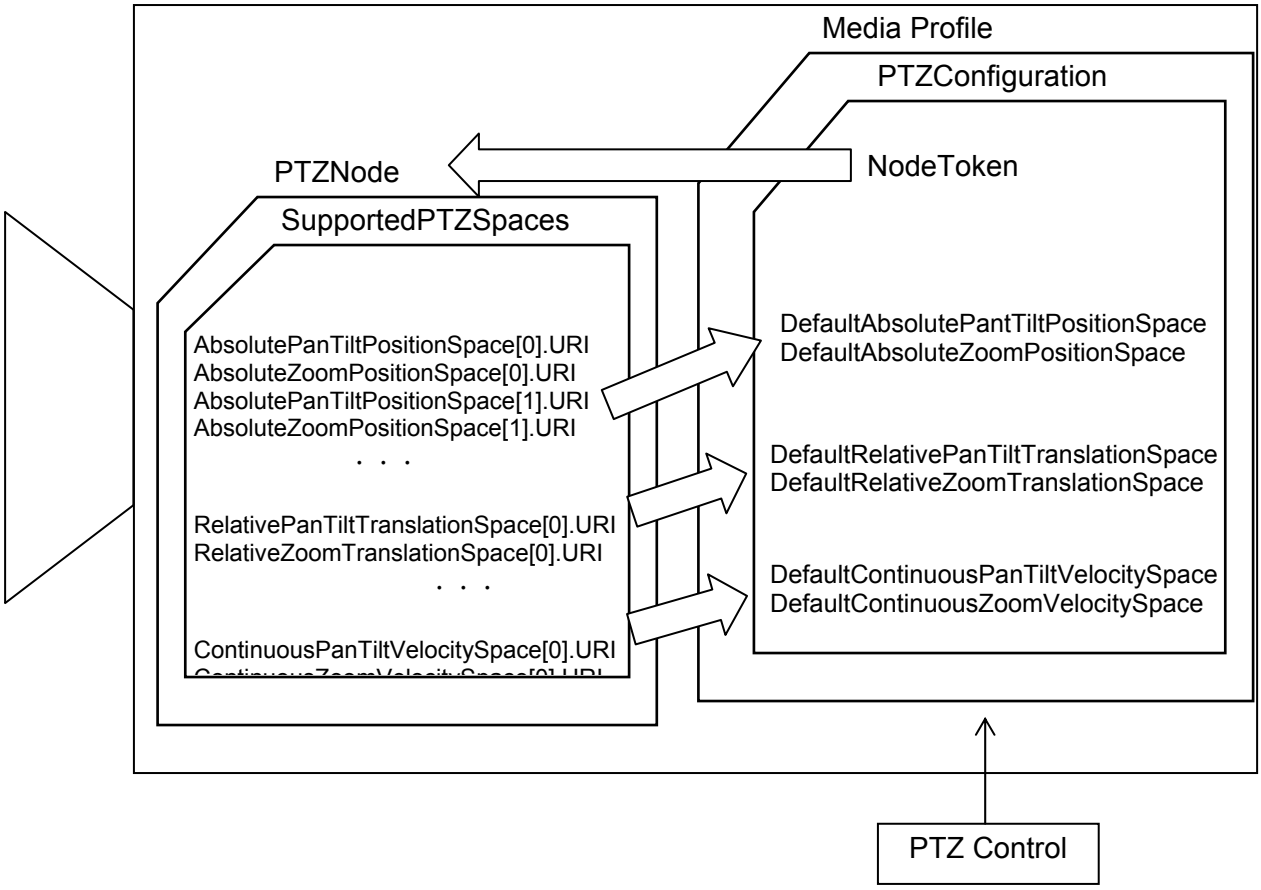
```
   // Setup stream configuration (RTP/RTSP) and the desired Profile:
   streamSetup.Stream = "RTP-Unicast";
   // Set Transport.Protocol to:
   // UDP for RTP/UDP
   // RTSP for RTP/RTSP (over TCP)
   // HTTP for RTP/RTSP/HTTP
   streamSetup.Transport.Protocol = "RTSP";
   streamSetup.Transport.Tunnel = null;

   // Get stream URI
   mediaUri = media.GetStreamUri(StreamSetup = streamSetup,
                                 ProfileToken = "metadata");
   // Fetch the RTSP stream, and handle the meta data
   App.fetch_metadata_rtsp_stream(mediaUri.Uri);
} // test_meta_data_streaming
```

7.6.5 RSTP-Communication Trace

[ONVIF/Realtime-Streaming]

8 Controlling

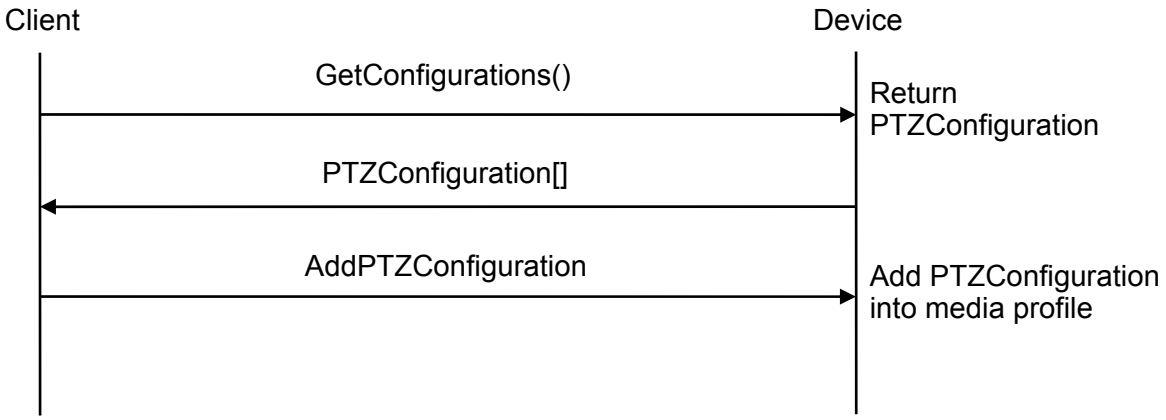
This chapter describes how to control the PTZ.



A PTZ-capable NVT may have one or many PTZ nodes. The PTZ node may be a mechanical PTZ driver, an uploaded PTZ driver on a video encoder, or a digital PTZ driver. The PTZ node is the lowest level entity of the PTZ Control, and it specifies the supported PTZ capabilities. `PTZConfiguration` has a node token and default settings which are indicated by URI. A `PTZConfiguration` is added in the media profile. Therefore, we can control PTZ operation through the media profile.

8.1 Adding a PTZ Configuration into a Media Profile

This use case describes how to add a new PTZ configuration into a specific media profile. The PTZ configuration cannot be added to default media profiles, so you must add the PTZ configuration before attempting a PTZ operation. The new PTZ configuration can be verified by calling `GetProfiles` or `GetProfile` in the media service.



8.1.1 Prerequisites

- `Profile1` must exist, which is a profile token with `PTZConfiguration`.

8.1.2 Targeted Services and Technologies

- [ONVIF/PTZ] and [ptz.wsdl]

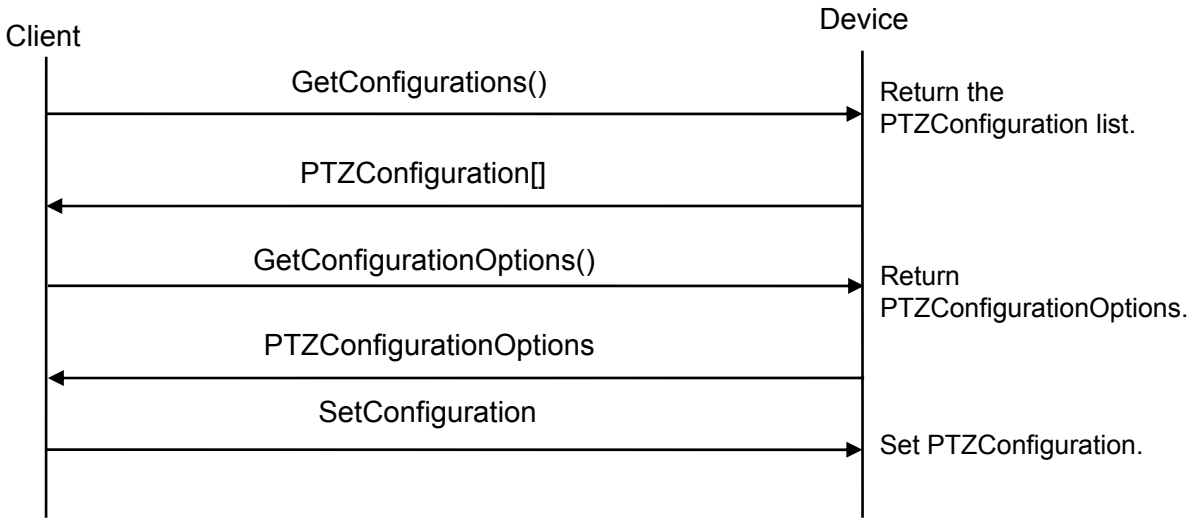
8.1.3 ONVIF::AddPTZConfiguration

This example describes how to add a PTZ configuration to a specific media profile. First, you must create the PTZ object. Then, get the PTZ configuration lists and select the top of the lists. Use the media profile token and PTZ configuration token to call `AddPTZConfiguration`.

```
// create the PTZ object to use the service
ptzService = getPTZService(MyPTZServiceAddress);

// get all PTZ configurations
ptzConfigurationsList = ptzService.GetConfigurations();

// use the first configuration and PTZConfigurations have at least one
ptzConfigurationToken = ptzConfigurationsList[0].token;

// add PTZ configuration into a certain Media profile
media.AddPTZConfiguration("Profile1", ptzConfigurationToken);

// now PTZ unit can be moved as shown in chapter 8.3
```

8.2 Changing a PTZ Configuration

This use case describes how to change a PTZ configuration. The PTZ service provides absolute move, relative move, and continuous move operations. This enables you to change PTZ control operations easily.



8.2.1 Prerequisites

None.

8.2.2 Targeted Services and Technologies

- [ONVIF/PTZ] and [ptz.wsdl]

8.2.3 ONVIF::ChangePTZConfiguration

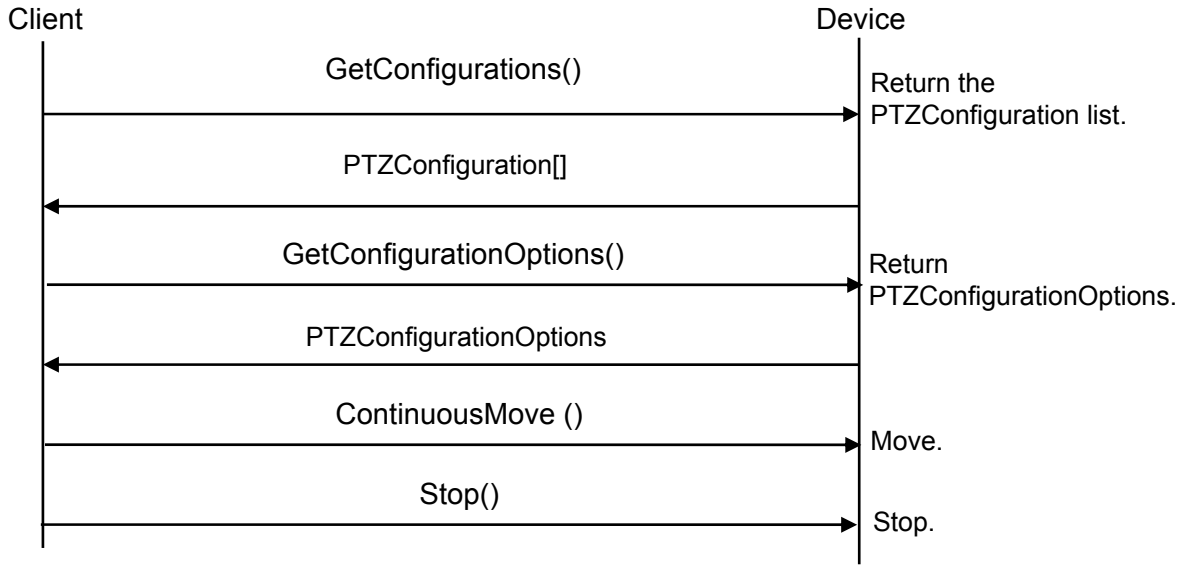
This example describes how to change the absolute PTZ method. At a high level, the process involves:

1. Creating the PTZ object to use the service
2. Getting the PTZ configuration lists and copying `PTZConfiguration` from `PTZConfigurationsList`
3. Setting the absolute PTZ in `PTZConfiguration` to the default URI and `PTZSpeed` from `ConfigurationOptions`
4. Using `PTZConfiguration` and `forcePersistence` to call `SetConfiguration`

```
//create the PTZ object to use the service
ptzService = getPTZService(MyPTZServiceAddress);

//get all PTZ configurations
ptzConfigurationsList = ptzService.GetConfigurations();

//copy from PTZ configuration list to PTZ congituration
ptzConfiguration = ptzConfigurationsList[0];

//get PTZ configuration options.
ptzConfigurationOptions =
ptzService.GetConfigurationOptions(ptzConfiguration.token);

if(ptzConfigurationOptions.Spaces.AbsolutePanTiltPositionSpace.size()>=2){
  ptzConfiguration.DefaultAbsolutePantTiltPositionSpace =
  ptzConfigurationOptions.Spaces.AbsolutePanTiltPositionSpace[1].URI;
}

if(ptzConfigurationOptions.Spaces.AbsoluteZoomPositionSpace.size()>=2){
  ptzConfiguration.DefaultAbsoluteZoomPositionSpace     =
  ptzConfigurationOptions.Spaces.AbsoluteZoomPositionSpace[1].URI;
}

if(ptzConfigurationOptions.Spaces.PanTiltSpeedSpace.size()>=2){
  ptzConfiguration.DefaultPTZSpeed.PanTilt.x =
  ptzConfigurationOptions.Spaces.PanTiltSpeedSpace[1].XRange.Max;
  ptzConfiguration.DefaultPTZSpeed.PanTilt.y =
  ptzConfigurationOptions.Spaces.PanTiltSpeedSpace[1].YRange.Max;
  ptzConfiguration.DefaultPTZSpeed.PanTilt.space =
  ptzConfigurationOptions.Spaces.PanTiltSpeedSpace[1].URI;
}

if(ptzConfigurationOptions.Spaces.ZoomSpeedSpace.size()>=2){
  ptzConfiguration.DefaultPTZSpeed.Zoom.x =
  ptzConfigurationOptions.Spaces.ZoomSpeedSpace[1].XRange.Max;
  ptzConfiguration.DefaultPTZSpeed.Zoom.space =
  ptzConfigurationOptions.Spaces.ZoomSpeedSpace[1].URI;
}

forcePersistence = false;
ptzService.SetConfiguration(ptzConfiguration, forcePersistence);

//now PTZ unit can be moved as shown in chapter 8.3
```

8.3 Move Operation

This use case describes how to move the PTZ unit.



8.3.1 Prerequisites

- `Profile1` must exist, which is a profile token with `PTZConfiguration`.

8.3.2 Targeted Services and Technologies

- [ONVIF/Media] and [media.wsdl]
- [ONVIF/PTZ] and [ptz.wsdl]

8.3.3 ONVIF::MoveControl

This example describes how to move the PTZ unit continuously. The process involves:

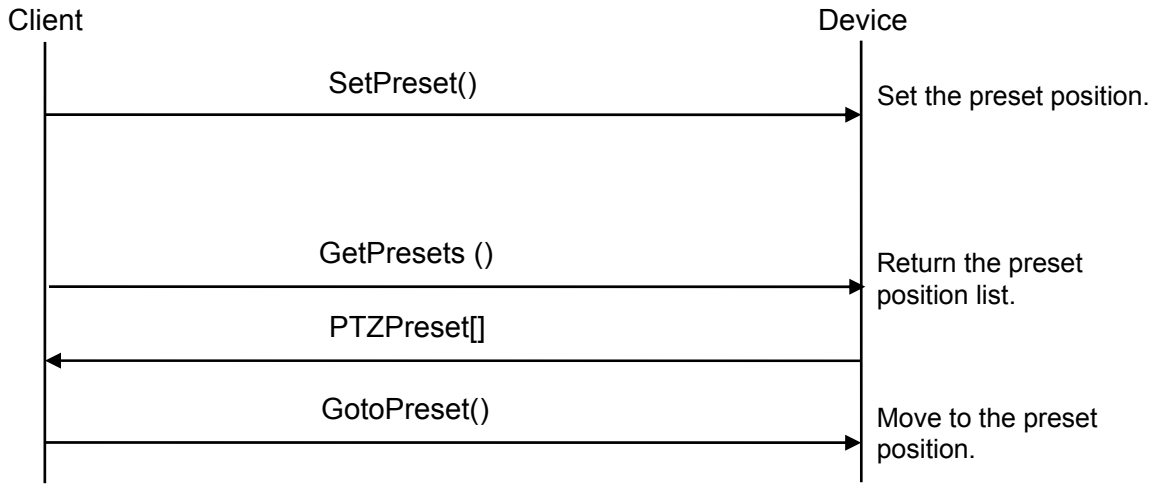
1. Creating the PTZ object to use the service
2. Getting the `PTZConfigurationOptions` using the token in the target media profile
3. Setting velocity using the PTZ node data
4. Starting a continuous move and stopping it after a specified time

```
// create the media object to use the service
mediaService = getMediaService(MyMediaServiceAddress);

// create the PTZ object to use the service
ptzService = getPTZService(MyPTZServiceAddress);


// get target profile
mediaProfile = mediaService.GetProfile("Profile1");


// get PTZ configuration options for getting continuous move range
ptzConfigurationOptions =
ptzService.GetConfigurationOptions(mediaProfile.PTZConfiguration.token);

// set velocity using PTZ configuration options data
velocity.PanTilt.x     =
ptzConfigurationOptions.Spaces.ContinuousPanTiltVelocitySpace[0].XRange.Max;
velocity.PanTilt.y     =
ptzConfigurationOptions.PTZSpaces.ContinuousPanTiltVelocitySpace[0].YRange.Max;
velocity.PanTilt.space =
ptzConfigurationOptions.PTZSpaces.ContinuousPanTiltVelocitySpace[0].URI;
velocity.Zoom.x        =
ptzConfigurationOptions.PTZSpaces.ContinuousZoomVelocitySpace[0].XRange.Max;
velocity.Zoom.space    =
ptzConfigurationOptions.PTZSpaces.ContinuousZoomVelocitySpace[0].URI;

// start continuous move
ptzService.ContinuousMove("Profile1", velocity );

// wait a certain time
Sleep(5000);

// stop continuouse move
ptzService.Stop("Profile1");
```

For more information about obtaining configuration options, see Section 8.2, Changing a PTZ Configuration.

8.4 Set / Goto Preset Position

The preset function can save the current device position parameters. If we set the preset position, the device can move there. Preset operations are set according to the media profile. This use case describes how to set a `/goto` preset position.



8.4.1 Prerequisites

- `Profile1` must exist, which is a profile token with `PTZConfiguration`.

8.4.2 Targeted Services and Technologies

- [ONVIF/PTZ] and [ptz.wsdl]

8.4.3 ONVIF::PresetControl

This example describes how to set the current position as a preset one. The process involves:

1. Calling `SetPreset`. This function needs an arbitrary name if it is a new preset position.
2. Using `GetPreset` to get the preset list which has already been added.
3. Calling `GotoPreset` to move the preset position to the top of the preset list.

```
// create the PTZ object to use the service
ptzService = getPTZService(MyPTZServiceAddress);

// set preset1
ptzService.SetPreset("Profile1", "PresetName1");

// get presets
ptzPresetsList = ptzService.GetPresets("Profile1");

// go to the first preset using PTZSpeed and PTZPresets have at least one
ptzService.GotoPreset("Profile1", PTZPresetsList[0].Name, PTZPresetsList[0].token,
NULL);
```

9 Eventing

The ONVIF specification includes three different types of event notifications:

- Real-time Pull-Point Notification Interface
- Basic Notification Interface (`WS-BaseNotification`)
- Notification Streaming Interface (metadata streaming)

晕啊，到底是GetServiceCapablities还是GetEventProperties决定采用哪种模型呢？

The following section describes the `GetEventProperties` action, which is a way of finding out what notifications a device supports and what information they contain. The next two sections describe how to set up the subscriptions for the two first methods above, and the last section describes how the Notification Message is processed. More information about the Notification Streaming Interface appears in Section 7.6, Setting Up Metadata Streaming.

9.1.1 GetEvent Properties

The `GetEventProperties` action returns the various event topics that the device supports.

9.1.2 Prerequisites

- A device must be discovered and the service URIs must be known.

9.1.3 Targeted Services and Technologies

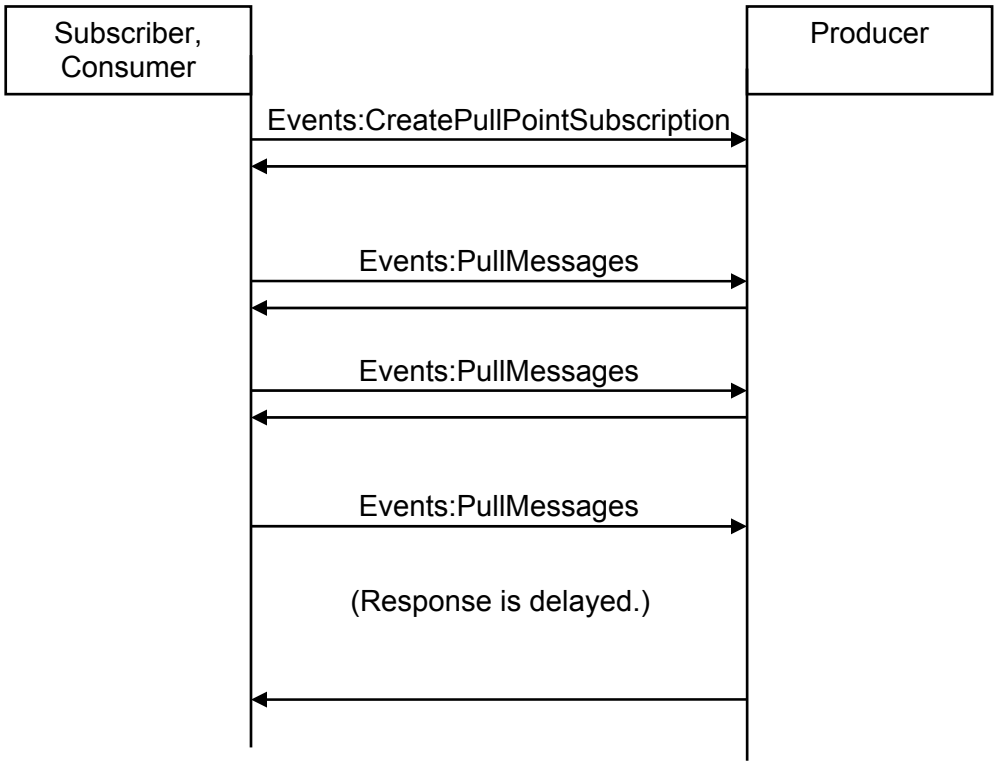
- [ONVIF/Event-Handling] and [event.wsdl]

9.1.4 ONVIF::TestGetEventProperties

```
test_GetEventProperties(onvifdev)
 {
   events = getEventService()onvifdev.eventsxaddr);
   // Fetch EventProperties
   // In the response we can find what filterdialect that are supported
   // and what Topics that the device support.
   // There may be both standardized and vendor specific topics.
   // Some SOAP Headers need to be set:
   events.Header.Action =
"http://www.onvif.org/ver10/events/wsdl/EventPortType/GetEventPropertiesRequest";
    events.Header.To = onvifdev.eventsxaddr;

   // Header set in events.Header above will be used.
   eventProperties = events.GetEventProperties();
   // TODO: Do something with the result

} // test_GetEventProperties
```

9.2 Setting Up PullPoint Subscription

`PullPoint` subscription is used when a client wants to fetch event notifications from a service. This is an ONVIF extension to the standard WS-BaseNotification mechanisms. First, a subscription is created and a `subscriptionReference` is returned, which is used in `PullMessages` requests to fetch the actual event notifications. If no notifications are available, the response is delayed.



9.2.1 Prerequisites

- A device must be discovered and the service URIs must be known.

9.2.2 Targeted Services and Technologies

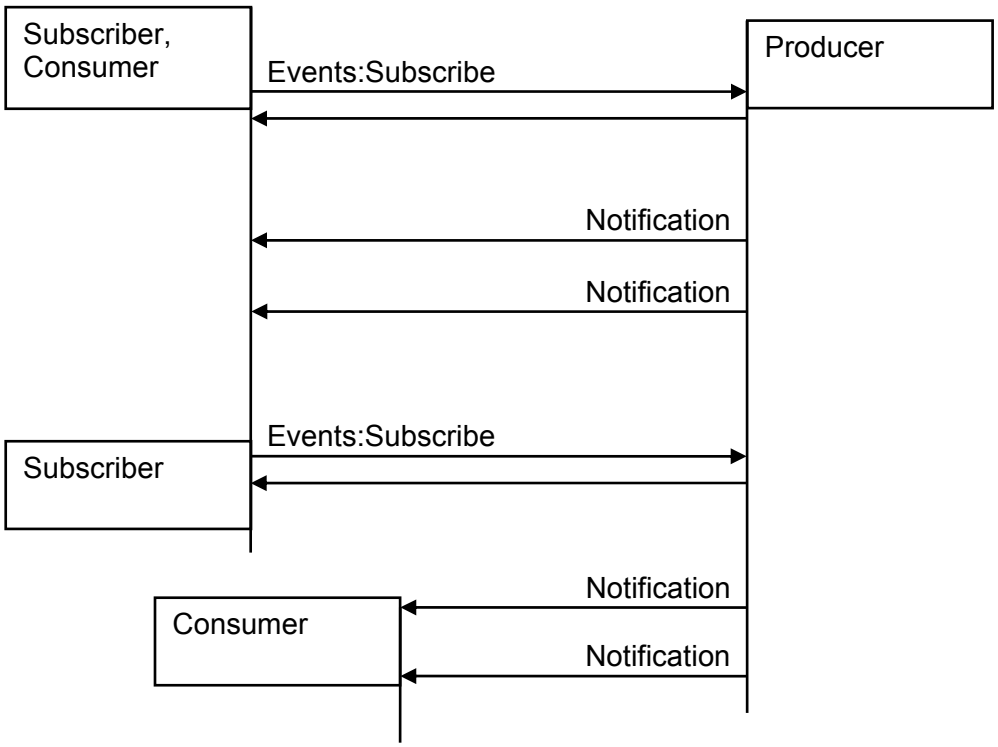
- [ONVIF/Event-Handling] and [event.wsdl]

9.2.3 ONVIF::TestPullPointSubscription

```
// Test ONVIF PullPoint subscription:
 // Setup filter and call events:CreatePullPointSubscription()
 // Call events:PullMessages() to fetch events.
 //
 test_pull_point_subscription(onvifdev)
 {
   events = getEventService(onvifdev.eventsxaddr);

  // Some SOAP Headers need to be set:

   events.Header.Action =
"http://www.onvif.org/ver10/events/wsdl/EventPortType/CreatePullPointSubscriptionReque
st";
   events.Header.To = onvifdev.eventsxaddr;

   // Support for SubscriptionPolicy is optional and we should check the Capabilities
   // before using it. In this example we do not specify any SubscriptionPolicy.
   // [ONVIF/Event handling, chapter 15]
   filter = Filter(TopicExpression = "tns1:Device//.");
   resp = events.CreatePullPointSubscription(Filter = filter,
                                             InitialTerminationTime = "PT1M",
                                             SubscriptionPolicy = NULL);
   // Some SOAP Headers need to be set:
   events.Header.Action =
"http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/PullMessagesRequest";
   events.Header.To = resp.SubscriptionReference.Address;

   // Copy ReferenceProperties and ReferenceParameters fields to header
   // See http://www.w3.org/Submission/ws-addressing/
   events.Header.ReferenceProperties =
     resp.SubscriptionReference.ReferenceProperties;
   events.Header.ReferenceProperties  =
     resp.SubscriptionReference.ReferenceParameters;

   // In case the device does not give us the requested InitialTerminationTime,
   // we check the response.
   timeout = App.CalcDuration(resp.TerminationTime,resp.CurrentTime);
   timeout = App.MinPeriod(timeout, "PT5S");

   // Issue a couple of PullMessages requests, bail out if no more events
   // We currently only process 1 event at the time so set MessageLimit to 1

   cnt = 30;
   while (cnt--) {
     // The PullMessages request does not get a response until timeout
     // or event arrives. (The events.Header is used as well)
     // The device itself should still be able to handle other requests
     // in the meantime.
     // Timeout says how long we want to wait before getting a response and
     // MessageLimit tells how many messages we want to wait for.
     aPullMessagesResponse = events.PullMessages(Timeout = timeout,
                                                 MessageLimit = 1);
     if (aPullMessagesResponse.NotificationMessage) {
       process_notification(aPullMessagesResponse.NotificationMessage);
     }
   } // while
 } // test_pull_point_subscription
```

9.3 Setting Up WS-BaseNotification

`WS-BaseNotification` is the standard WS Notification mechanism. A subscription is set up, and the service handling the notification connects to the specified URL and POST the Notification message. The connection on which the Notification is sent is initiated by the producer, and the consumer does not need to be the same entity that sets up the subscription.



9.3.1 Prerequisites

- A device must be discovered and the service URIs must be known.

9.3.2 Targeted Services and Technologies

- [ONVIF/Event-Handling] and [event.wsdl]

9.3.3 ONVIF::TestNotificationSubscription

```
// Test notification subscription
 //
 // WS-BaseNotification using NotificationConsumer interface
 // http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf
 //
 test_notification_subscription(onvifdev)
 {
   events = ONVIF::Events(onvifdev.eventsxaddr);
   // The following could possibly be hidden by the above depending
   // on framework used
   events.Header.Action =
     "http://docs.oasis-open.org/wsn/bw-2/NotificationProducer/SubscribeRequest";
   evenst.Header.To = onvifdev.eventsxaddr;

   // We supply an EndPointReference for the consumer - see [WS-Addressing],
   // where the device can Notify us by POST:ing Notify messages
   // We set up a listener on a TCP port to receive notifications and
   // act as a webserver (or you could instead point it to an
   // existing webserver - even on another host)

   consumerfd = net_listener_setup(App.GetNotificationPort);
   consumerReference =
     wsnt::ConsumerReference(Address = App.GetNotificationUrl());

   // Subscribe to the events, the events.Header is used as well.
   events.Subscribe(ConsumerReference = consumerReference,
                    InitialTerminationTime = "PT1M")
   // What a while for responses
   while (new_connection_and_no_timeout(consumerfd)) {
     newfd = accept(consumerfd);
     notify = read_all(newfd);
     process_notification(notify.NotificationMessage);
   }
 } // test_notification_subscription
```

9.4 Processing NotificationMessage

The notification message looks the same regardless of the method by which it was delivered. The following sections provide a simple example of how such a message can be processed. The content of a Notification can be vendor-specific.

9.4.1 ONVIF::ProcessNotificationMessage

This example uses the same processing, whether it is a `PullMessagesResponse` containing one or more `NotificationMessage`, or a Notification containing a single `NotificationMessage`.

This function processes a single `NotificationMessage`, so it should be called multiple times for each of the messages in a `PullMessagesResponse`.

```
// Parses/Processes a NotificationMessage
process_notification(theNotificationMessage) {
   // For ONVIF devices UtcTime is a required attribute to the
   // NotificationMessage/Message/Message tag
   utctime = theNotificationMessage.Message.Message.UtcTime;
   // The optional PropertyOperation attribute tells if the notification
   // is due to that something has changed or just to inform about the state.
   // Valid values are: Initialized, Deleted and Changed.
   op = theNotificationMessage.Message.Message.PropertyOperation;

   // Get the topic, the dialect and the producer
   topic = theNotificationMessage.Topic;
   topic_dialect = theNotificationMessage.Topic.Dialect; // Attribute
   producer = theNotificationMessage.ProducerReference.Address;

   // For WS-BaseNotification the Header contains To and Action and
   // any ReferenceParameters specified in the request
   // This example does not use any of those though.

   // tt:Message may contain Source, Key, Data of type ItemList
   // and Extension of anyType.
   // ItemList (Source, Key and Data) is recommended to contain
   // tt:SimpleItem elements with Name and Value attributes
   // but it could be the more complex ElementItem as well.
   //
   // The actual content of each Topic is described in the response to
   // the events:GetEventProperties function.
   //
   // This example only handles one SimpleItem in each of
   // Source, Key and Data - but there could be multiple items.
   sourceList = theNotificationMessage.Message.Message.Source;
   // Get the Name and Value attributes from the SimpleItem.
   source_name  = sourceList[0].SimpleItem.Name;
   source_value = sourceList[0].SimpleItem.Value;
   keyList = theNotificationMessage.Message.Message.Key;
   key_name  = keyList[0].SimpleItem.Name;
   key_value = keyList[0].SimpleItem.Value;
   dataList = theNotificationMessage.Message.Message.Data;
   data_name  = dataList[0].SimpleItem.Name;
   data_value = dataList[0].SimpleItem.Value;

   // Process the event data (application specific)
   App.handle_notification(topic, topic_dialect, producer,
                           source_name, source_value,
                           key_name, key_value, data_name, data_value);
 } // process_notification
```

10 Storage

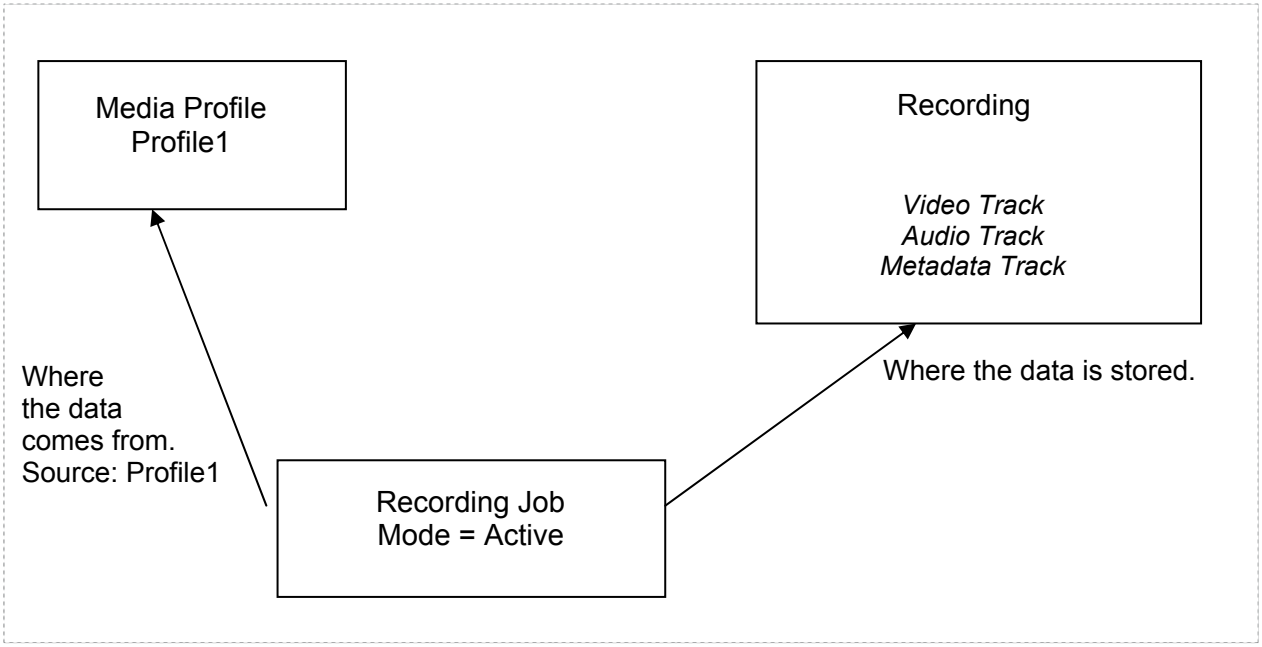
10.1 Starting a Local Recording

This use case demonstrates how to start a local recording on a device. The device has embedded storage (such as an SD card) to store the data. The client has already set up a media profile on the device with a token `Profile1` that should be used for recording.

First, the client asks for the existing recordings on the storage unit. In this example, the client uses an existing recording and possibly overwrites or adds new data. A new recording could also be created, if this is supported by the device.

Next, the client changes the configuration of the recording. It stores the necessary information about the source (like the name, location, or IP address of the device), the description of the content, and the retention time.

Then, it creates a `RecordingJob` that transfers the data from the `RecordingSource` (in this use case, the media profile) to the recording. The Recording Job mode is set to `Active`, so the device starts the recording automatically and no interaction from the client is necessary.



10.1.1 Prerequisites

- The client has already configured a profile with the token `Profile1` to use for recording.
- There is a recording that contains the necessary tracks.

10.1.2 Targeted Services and Technologies

- Recording Service: see [ONVIF/Recording] and [recording.wsdl]

10.1.3 ONVIF::StartLocalRecording

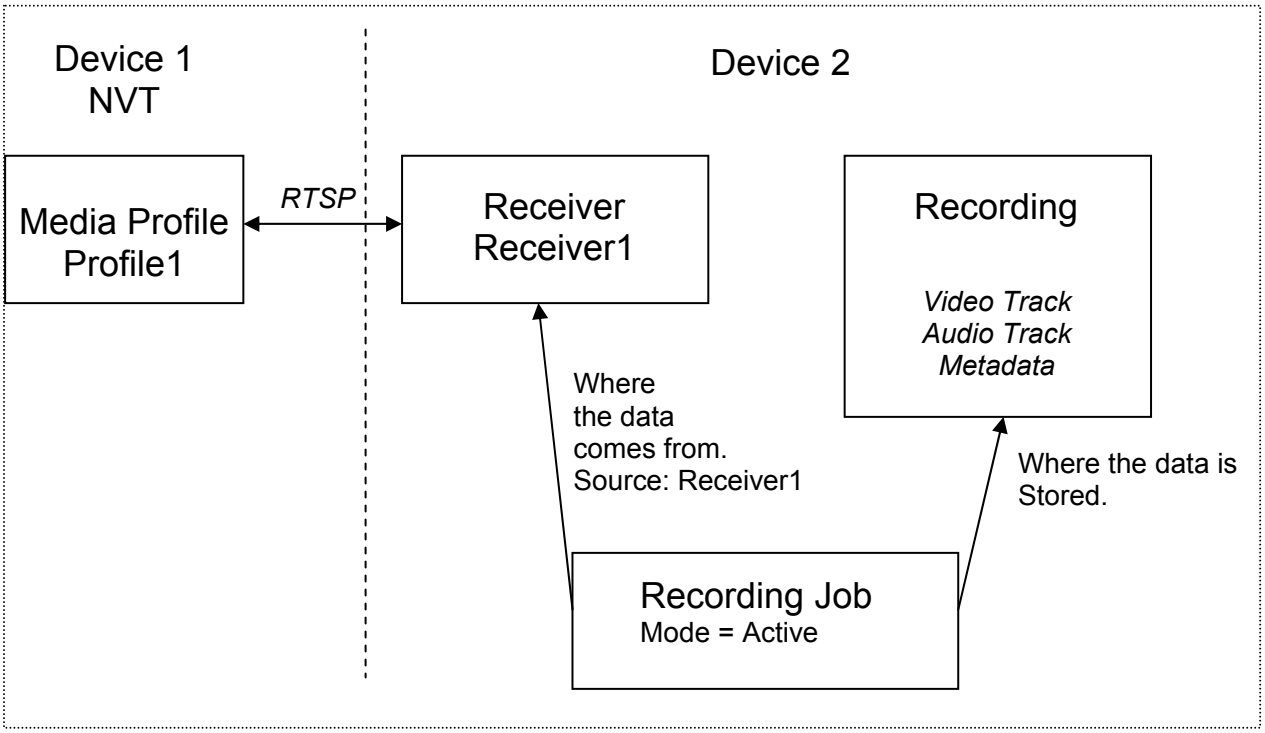
This example shows the steps that are required for setting up a local recording.

```
// create the needed recording object using RecordingServiceAddress that can be
// requested using the GetCapabilities command
recordingService    = getRecordingServiceService(MyRecordingServiceAddress);

// request existing recordings
recs = recordingService.GetRecordings();

//we assume that all recordings are currently unused. Select the first one. We
//reconfigure the recording for our needs

MyRec = recs[0];

//set the RecordingConfiguration; these values are set by the client and stored in
the
//device; the RecordingConfigurationSource gives information about the source of the
//recording

RecordingConfiguration.Source.SourceID      = "Device 1";
RecordingConfiguration.Source.Name          = "camera PT677X";
RecordingConfiguration.Source.Location      = "Room 1";
RecordingConfiguration.Source.Description   = "continuous recording of room 1";
RecordingConfiguration.Source.Address       = "192.168.0.2";

RecordingConfiguration.Content              = "Recording from device 1";
RecordingConfiguration.MaximumRetentionTime = "PT0S";

//now the recording can be configured
recordingService.SetRecordingConfiguration(MyRec.token,RecordingConfiguration);



//now we have to create a RecordingJob; set the RecordingJobConfiguration recording
//source is set to "Profile1"
//We set the mode to "Active" to start the recording immediately
RecordingJobConfiguration.RecordingToken = MyRec.token;
RecordingJobConfiguration.Mode           = "Active";
RecordingJobConfiguration.Priority       = 1;
RecordingJobConfiguration.SourceToken.Token = "Profile1";
RecordingJobConfiguration.SourceToken.Type  =
     "http://www.onvif.org/ver10/schema/Profile";
RecordingJobConfiguration.AutoCreateReceiver = false;



//create the recording job and start the recording
recJobToken = RecordingService.CreateRecordingJob(RecordingJobConfiguration);
```

10.2 Starting a Recording from a Remote Device

This use case shows how to setup a remote recording from another camera in the network. Therefore the client has configured a media profile on the remote device and has requested the stream URL.

The client has also already created a recording (`MyRec`) that should be used to store the data. It sets up a Recording Job that transfers the data from the receiver to the recording. The `RecordingJob` has an `AutoCreateReceiver`. If this flag is set to true, a Receiver is automatically created in the Receiver Service and is associated with the recording job. If the recording job is deleted this receiver will also be deleted without client interaction. The client has to configure the receiver with the already known RTSP URI and the stream setup. Then it can start the recording job by setting the recording job mode to `Active`.



10.2.1 Targeted Services and Technologies

- [ONVIF/Receiver] and [receiver.wsdl]
- [ONVIF/Recording] and [recording.wsdl]

10.2.2 ONVIF::StartRemoteRecording

This example shows the steps that are necessary for setting up a remote recording from an ONVIF transmitter device.

```
// create the needed recording and receiver object using the RecordingServiceAddress
// and the ReceiverServiceAddress that can be requested using the GetCapabilities
// command
recordingService    = getRecordingService(MyRecordingServiceAddress);
receiverService     = getReceiverService(MyReceiverServiceAddress);

//now we have to create a RecordingJob; set the RecordingJobConfiguration recording
//we set the AutoCreateReceiver Flag. The device will create a receiver and
associates
// it automatically with the RecordingJob. The mode is set to idle, because we have
to
// configure the receiver first, before we can start recording
RecordingJobConfiguration.RecordingToken = MyRec.token;
RecordingJobConfiguration.Mode           = "Idle";
RecordingJobConfiguration.Priority       = 1;
RecordingJobConfiguration.Source.SourceToken        = null;
RecordingJobConfiguration.Source.AutoCreateReceiver = true;


//create the recording job
RecordingJobConfiguration.RecordingToken = MyRec.token;
JobToken, JobConfiguration =
 recordingService.CreateRecordingJob(RecordingJobConfiguration);

//set the receiver, assume that we have a valid stream uri from device we want to
// record the data from. This stream uri can be retrieved using the remotes devices
// media service

//the device creates a receiver e.g with token "Receiver1"
ReceiverConfiguration.Mode     = "AlwaysConnect";
ReceiverConfiguration.MediaUri = MyUri;
ReceiverConfiguration.StreamSetup.Stream = "unicast";
ReceiverConfiguration.StreamSetup.Transport.Protocol = "UDP";
ReceiverConfiguration.StreamSetup.Transport.Tunnel   = null;

ReceiverService.ConfigureReceiver("Receiver1",ReceiverConfiguration);


//set the RecordingJob to active to start recording of data
recordingJob.SetRecordingJobMode(JobToken,"Active");
```

10.3 Finding a Recording

This use case describes how a simple search for recordings could be done. The client wants to create a list of available recording footage in a recording. This list could be used to replay the data and give information about which events happened during this time.

Id	StartTime	StopTime	Events
1	2010-09-27 09:30:21.21	2010-09-27 09:35:24.25	Motion
2	2010-09-27 09:36:43.45	2010-09-27 09:42:28.32	Motion
3	2010-09-27 09:51:22.22	2010-09-27 10:02:01.01	Motion
4	2010-09-27 10:12:54.03	2010-09-27 10:13:00.25	Motion
5	2010-09-27 10:13:00.25	2010-09-27 10:15:24.23	Motion

In this use case, the device has one recording with audio, video, and meta tracks. The client looks for the `IsDataPresent` event to find the times when a recording job was started and when it was stopped. Therefore, the client sets up a `FindEvents` job and waits for the results. Afterwards, it can go through the list and find the start and stop times of the recording job.

10.3.1 Prerequisites

- The device contains a recording.

10.3.2 Targeted Services and Technologies

- [ONVIF/Search] and [search.wsdl]
- [ONVIF/Recording] and [recording.wsdl]

10.3.3 ONVIF::FindRecording

This example shows the steps that are required for this use case.

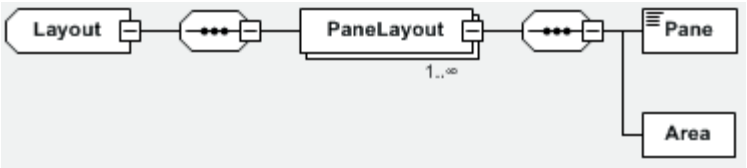
```
// create the needed recording object and the search object using
// RecordingServiceAddress and the SearchServiceAddress that can be requested
// using the GetCapabilities command
recordingService    = getRecordingService(MyRecordingServiceAddress);
searchService       = getSearchService(MySearchServiceAddress);

// then the client asks for the available recordings to get the recording token
GetRecordingsResponseItem RecsItem = recordingService.GetRecordings();

// in this example there is only one recording
MyRec = RecsItem[0].RecordingToken;
// create a event search, we are looking for the mandatory is dataPresent event
SearchScope.IncludedRecordings = MyRec;
SearchFilter.TopicExpression.Dialect
    ="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet";
SearchFilter.TopicExpression.any = "tns1:RecordingHistory/Track/State";
IncludeStartState = false;
MaxMatches = 100;
KeepAlive = PT1M;
StartPoint = GetCurrentUtcTime();
EndPoint = 2001-12-17T09:30:47.0Z;
// start a backward search. The start time is set to the current time; the end
//time is set to zero. The client wants to receive the last 100 events.
JobToken = searchService.FindEvents(
StartPoint,EndPoint,SearchScope,SearchFilter,IncludeStartState,MaxMatches,KeepAlive);

// call GetEventsSearchResult to get the search results. The GetEventSearchResult
// is an asynchrony command; it shouldn't block the search service or the client.
// If it is supported by the SOAP Framework the client sets up an asynchrony
```

```
// command
//The client uses a wait time of 1 minute (time that should be enough to complete the
//search
WaitTime = PT1M;
FindEventResultList = searchService.GetEventSearchResult(JobToken,WaitTime);

// now we have a list of Events including the times when the recording was stopped
// and started. We can go through this list and print the list for displaying
start = 0;
end   = 0;
ID = 0;

foreach(result in FindEventResultList)
{
    if( result.Event.Message.Data.Name  == "IsDataPresent" &&
        result.Event.Message.Data.Value == true)
    {
      start = result.Event.Message.UTCTime;
    }
    if( result.Event.Message.Data.Name  == "IsDataPresent" &&
        result.Event.Message.Data.Value == false)
    {
      end = result.Event.Message.UTCTime;
    }
    if(start && end)
    {
        print (ID,start,end);
        ID++;
        start = 0;
        end = 0;
    }
}
```

11 Display

This chapter focuses on display devices, which are devices that provide the Display service interface and functionality.

A display device provides video outputs which represent monitors or displays. A video output provides so-called *panes*. A pane is a region within the video output where a stream can be displayed after decoding. The pane is bound to the video output with its associated *layout*, which defines one or more regions to display. The structure holding the `PaneLayouts` is an ordered list. This is essential because with overlapping panes, the top elements in the list are displayed over the lower elements as shown in the following tree diagram of the Layout.



This structure provides the current layout of panes on the screen. The `PaneLayout` is a list of all panes visible on a `VideoOutput`. One pane is represented by a `PaneLayout` entity. The `Pane` parameter contains the reference to the associated `PaneConfiguration` and an `Area` object. The `Area` contains the values for top, bottom, right, and left, which describe the geometrical dimensions of the pane.

NOTICE: Make sure that you do not mix up the order when parsing or serializing the Layout structures, because the display might behave in an unexpected manner. For more information, see section 2 of [ONVIF/Display-Service] which describes Layout.

NOTICE: The area coordinate values are expressed in normalized units that range between -1.0 and 1.0. If two continuous display regions share the same border value with ranges that do not overlap, then they do not overlap at all. See the additional descriptions in [display.wsdl].

The following figure shows the relationship between video output, `Layout`, and panes.

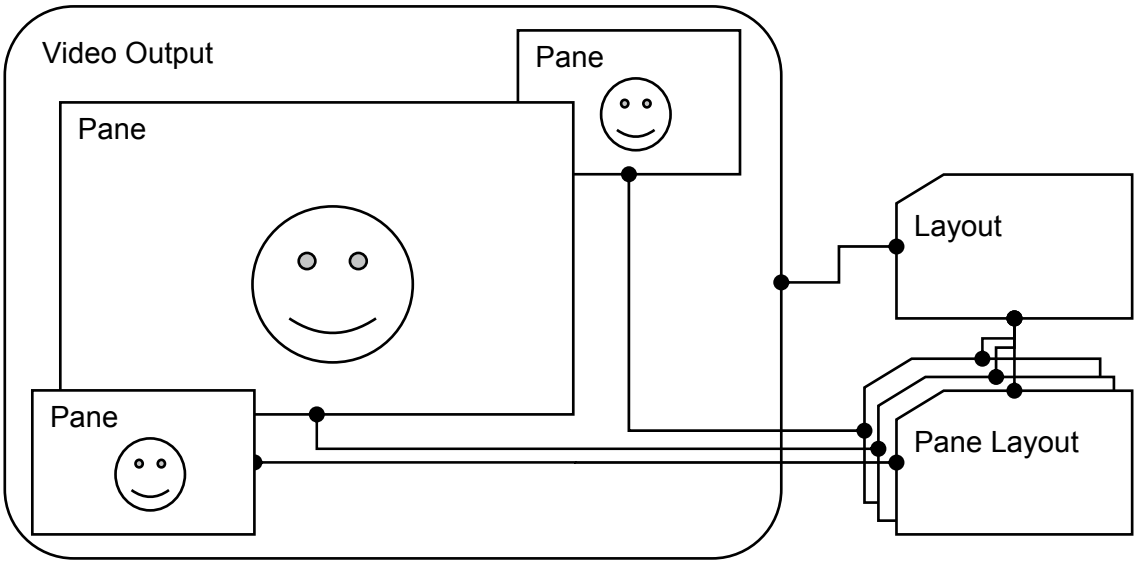


Figure 1: Video Output and Panes

Decoding on its own is strictly associated with a pane and the receiver assigned to that pane. The decoder itself doesn't provide any real parameters. Instead, it adapts to the received stream of the NVT automatically according to its capabilities. To ensure this will not fail, the NVT should be set up within the limits of the `CodingCapabilities` of a decoder entity. The following figure shows this relationship.

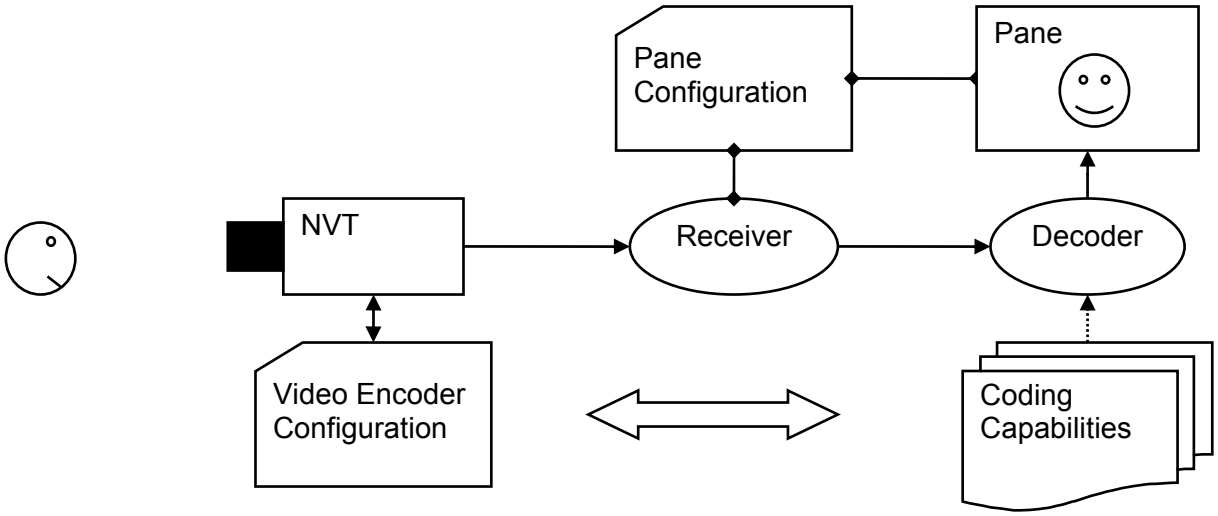
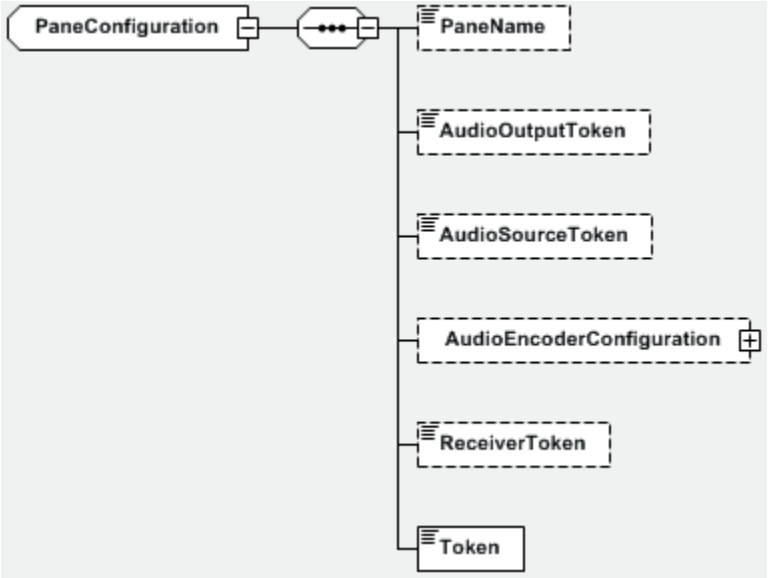


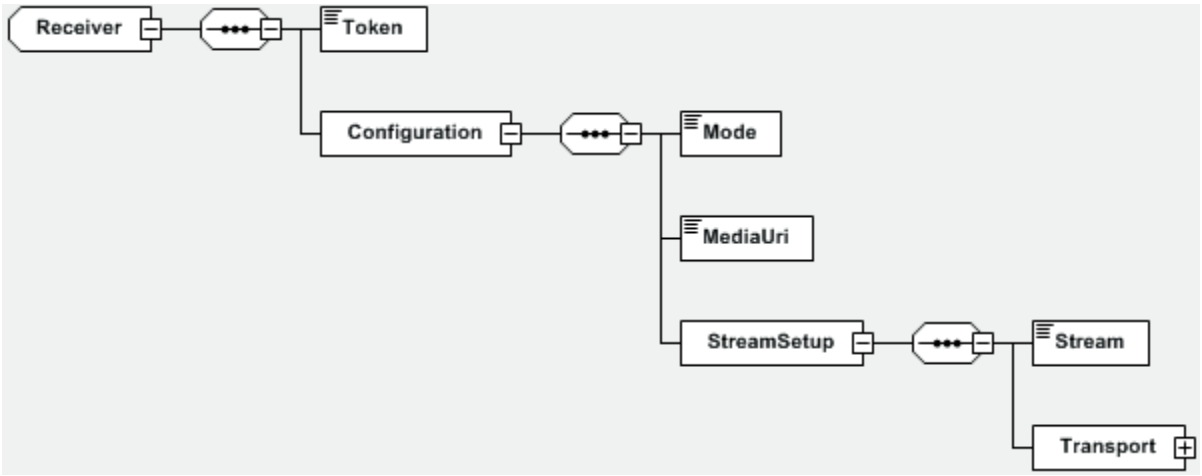
Figure 2: NVT, Receiver, and Pane

The corresponding structures are called `PaneConfiguration` and `Receiver`. The following section discusses `PaneConfiguration`, which binds a received stream to a pane.



The `Token` parameter is referenced by the `Pane` parameter in the `PaneLayout` items of a `Layout` structure. The `PaneConfiguration` entities are managed independently from the `Layouts` and can be seen as entities of decoders. Input for the decoder is provided by a `Receiver` instance, which is bound to a `PaneConfiguration` using the `ReceiverToken` parameter.

The following figure provides more information about the `Receiver` structure.



The `Token` parameter provides the required reference to the `Receiver`, which is referenced by the `ReceiverToken` parameter in the `PaneLayout` structure.

To attach a stream to a display, a configured receiver is placed as reference into a `PaneConfiguration`. That `PaneConfiguration` now must be associated with a pane in the current display layout.

11.1 Configuring a Display Device to Show a Stream

This section describes the general principles for configuring a device to display a desired stream from an NVT. It shows the basic technique and command order for working with a display device.

11.1.1 Prerequisites

- A configured `Receiver` (Section 11.4 provides background information)
- A free `PaneConfiguration` available in the Layout
- IP of device to configure (represented by the *`ip`* variable)

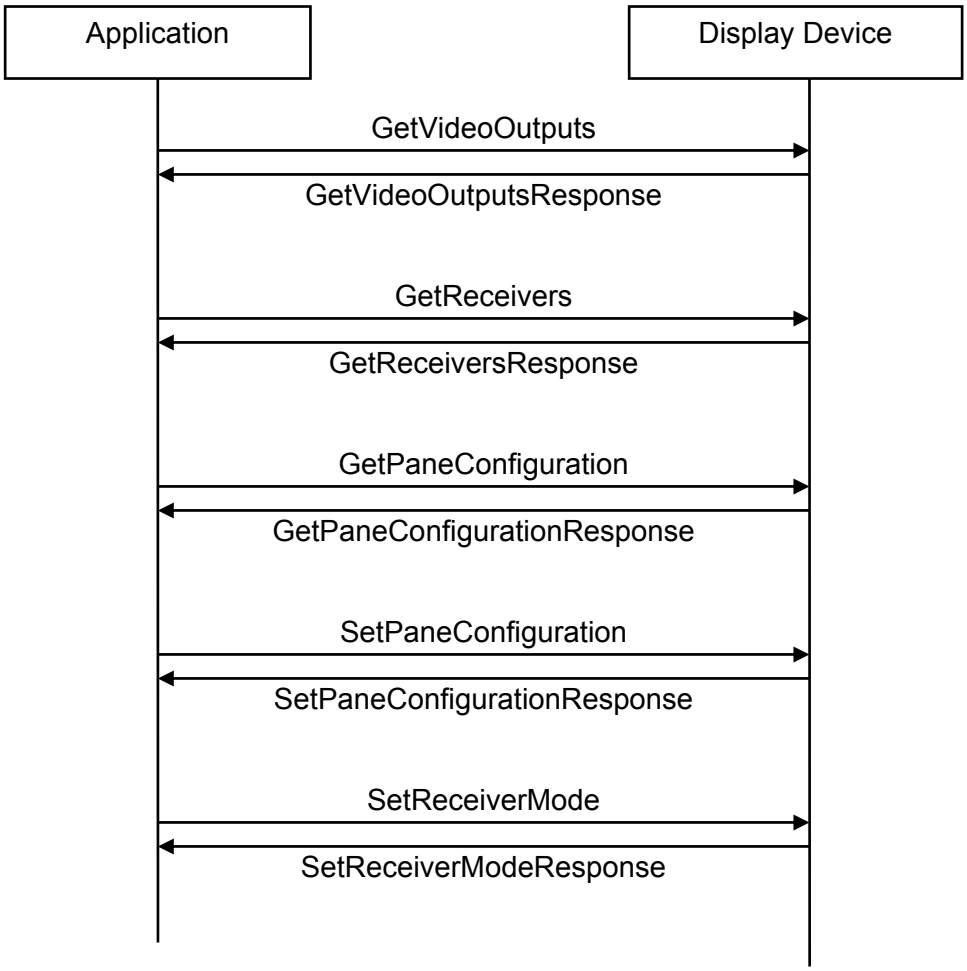
11.1.2 Targeted Services and Technologies

- Device IO service: see [ONVIF/Device-IO-Service] and [deviceio.wsdl]
- Display service: see [ONVIF/Display-Service] and [display.wsdl]
- Receiver service: see [ONVIF/Receiver-Configuration] and [receiver.wsdl]

11.1.3 ONVIF::AttachReceiverToPane

This example involves the following process:

1. The call endpoints for the services are initialized and a video output is selected.
2. The video output configuration is set up. This process is highly vendor- and device-dependent, and is therefore beyond the scope of this document.
3. The layout must be retrieved to obtain the current list of visible panes and their positions on the video output.
4. A pane can be selected and associated with a receiver.
5. The receiver is set to an active state to start the video stream.

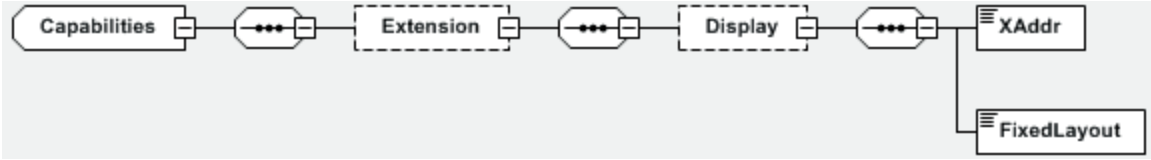


```
receiverService = getReceiverConfigurationService( ip );
deviceIOService = getDeviceIOService( ip );
displayService  = getDisplayService( ip );

// select a video output of a display device
// SOAP trace, see Annex B
videoOutputList = deviceIOService.GetVideoOutputs( );
videoOutput = App.selectVideoOutput( videoOutputList );

// here we select the first configured receiver
// SOAP trace, see Annex B
receiverList = receiverService.GetReceivers( );
receiver = App.selectReceiver( receiverList );

// now select PaneConfiguration for one of the panes in the desired layout
// Here you see one of two possibilities to get the Layout for a video output,
// see next chapter for other
paneLayout = App.selectPaneLayout(videoOutput.Layout.PaneLayout );
// SOAP trace, see Annex B
paneConfiguration = displayService.GetPaneConfiguration( videoOutput.token ,
paneLayout.pane );

//connect receiver to Pane
paneConfiguration.ReceiverToken = receiver.Token;
// SOAP trace, see Annex B
DisplayService.SetPaneConfiguration( videoOutput.token , paneConfiguration );

//start the receiver
receiverMode=ONVIF.tt.ReceiverMode( "AlwaysConnect" );
// SOAP trace, see Annex B
receiverService.SetReceiverMode( receiver.Token , receiverMode );
```

11.2 Creating and Deleting PaneConfiguration

This section describes the steps required to create a new `PaneConfiguration` and to connect it to the system. This process is based on the assumption that the targeted display device is capable of freely creating, configuring, and placing panes on a video output. This is signalled by the device in one of two ways:

- Within the Capabilities parameters for the Display service, as shown below.



  If the parameter `FixedLayout` is set to false, the display device has this capability. For an introduction on how to retrieve this information, see [ONVIF/Display-Service].

- Using the `GetDisplayOptions` interface. The return message provides `CodingCapabilities` and optionally `LayoutOptions`. If `LayoutOptions` is missing, the device also provides the capability to freely create and place the panes within a layout.

11.2.1 Prerequisites

- A display device capable of dynamically creating and deleting `PaneConfigurations`
- A receiver that is already configured (see Section 11.4)
- IP of device to configure (represented by the *`ip`* variable)
- IP of an NVT to include (represented by the *`nvt`* variable)
- The Pane

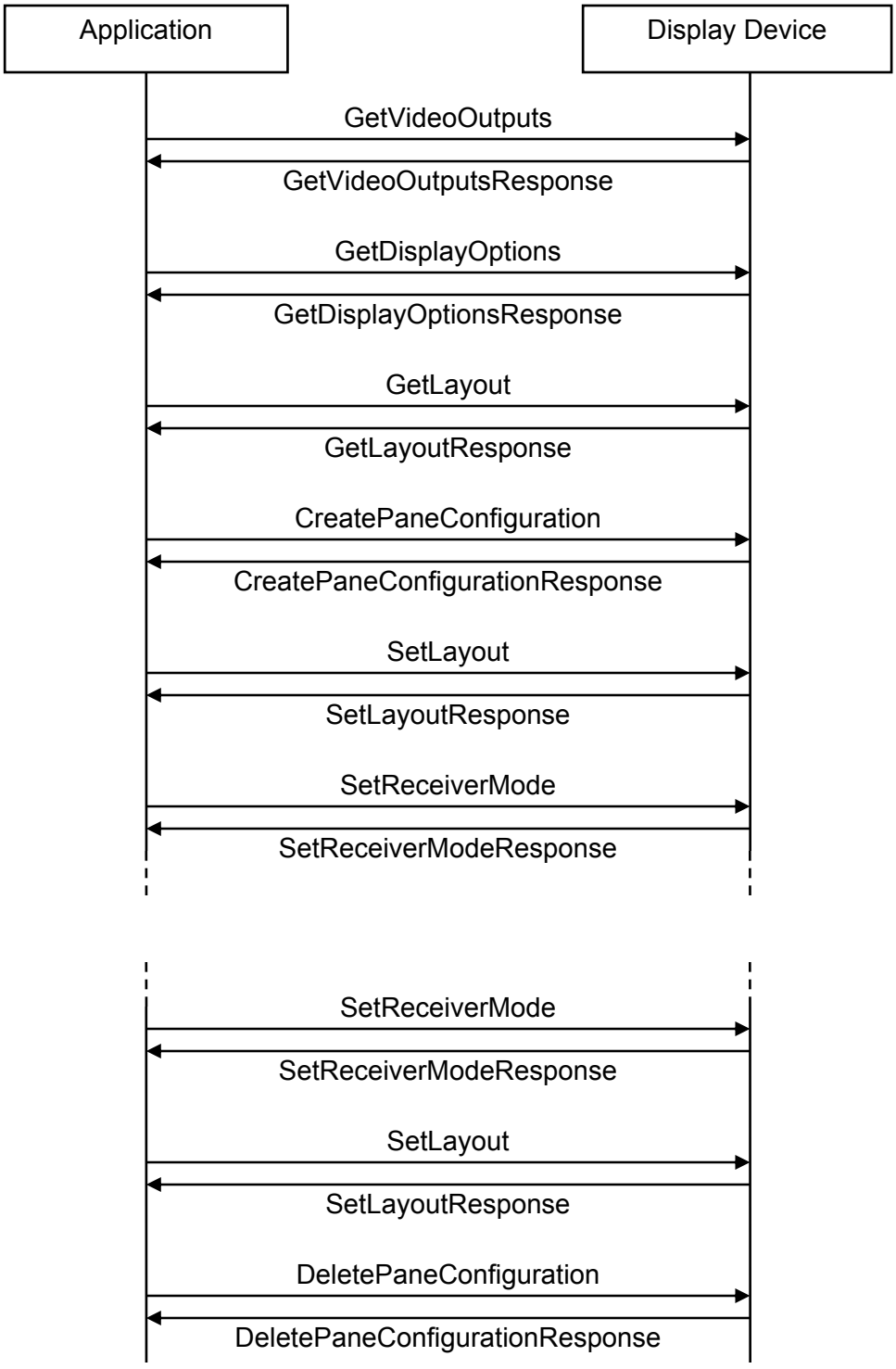
11.2.2 Targeted Services and Technologies

- Device IO service: see [ONVIF/Device-IO-Service] and [deviceio.wsdl]
- Display service: see [ONVIF/Display-Service] and [display.wsdl]
- Receiver service: see [ONVIF/Recevier-Configuration] and [receiver.wsdl]

11.2.3 ONVIF::CreateNewPaneConfiguration

This example involves the following process:

1. Getting the appropriate `VideoOutput` along with the receiver to display.
2. Getting a `paneLayout` by index to modify the list of `PaneLayout` objects in place.
3. Using the modified list to update the device configuration to establish the connection between the newly created `PaneConfiguration` and an existing `PaneLayout`.


Application
Display Device
GetVideoOutputs
GetVideoOutputsResponse
GetDisplayOptions
GetDisplayOptionsResponse
GetLayout
GetLayoutResponse
CreatePaneConfiguration
CreatePaneConfigurationResponse
SetLayout
SetLayoutResponse
SetReceiverMode
SetReceiverModeResponse
SetReceiverMode
SetReceiverModeResponse
SetLayout
SetLayoutResponse
DeletePaneConfiguration
DeletePaneConfigurationResponse

```
deviceIOService = getDeviceIOService( MyDeviceIOServiceAddress );
displayService  = getDisplayService( ip );
receiverService = getReceiverConfigurationService( ip );

// select a video output of a display device
// SOAP trace, see previous Annex B
videoOutputList = deviceIOService.GetVideoOutputs( );
videoOutput = App.selectVideoOutput( videoOutputList );

// check if device is supporting desired configuration mechanism
// SOAP trace, see Annex B
displayOptionsResponse = displayService.GetDisplayOptions( VideoOutput.token );

// Here you see the second possibility to get the Layout for a VideoOutput.
// for the first possibility see the chapter before
// SOAP trace, see Annex B
layout = displayService.GetLayout( VideoOutput.token );

if ( ! present( displayOptionsResponse.LayoutOptions ) )
{
    // We create a new receiver for this pane
    // See chapter 11.4.3 for details
    receiver = ONVIF::CreateReceiver( ip , nvt , VideoOutput.token );
    // now create PaneConfiguration for one of the panes in the desired layout
    paneConfiguration = tt.PaneConfiguration( );
    paneConfiguration.Token = App.createUniquePaneToken( );
    paneConfiguration.ReceiverToken = receiver.Token;
    // SOAP trace, see Annex B
    displayService.CreatePaneConfiguration( videoOutput.token , PaneConfiguration );

    // since the PaneConfiguration is not associated with
    // a pane in the layout, we have to create a new pane entry
    paneLayout = new PaneLayout();
    paneLayout.Pane = paneConfiguration.Token;
    paneLayout.Area.top= 1.0;
    paneLayout.Area.bottom=0.0;
    paneLayout.Area.left=0.0;
    paneLayout.Area.right=1.0;
    App.appendToList( layout.PaneLayouts , paneLayout );
    // SOAP trace, see Annex B
    displayService.SetLayout( VideoOutput.token , layout );

    // start the receiver
    receiverMode=ONVIF.tt.ReceiverMode( "AlwaysConnect" );
    // SOAP trace, see Annex B
    receiverService.SetReceiverMode( receiver.Token , receiverMode );

    // just waist some time before cleaning up
    App.waitSomeTime( );

    // stop the receiver and remove pane configuration
    receiverMode=ONVIF.tt.ReceiverMode( "NeverConnect" );
    receiverService.SetReceiverMode( receiver.Token , receiverMode );
    App.removeFromList(layout.PaneLayouts,paneLayout);
    displayService.SetLayout( VideoOutput.token , layout );
    // SOAP trace, see Annex B
    displayService.DeletePaneConfiguration( videoOutput.token , PaneConfiguration );
}
```

11.3 Changing the Layout Based on LayoutOptions

This use case describes how to change the layout of a video output when the display device does not support dynamic creation and deletion of pane entities.

To change the layout on such a device, more information is required: the list of possible layouts that a video output can arrange. This is provided by the optional `LayoutOptions` structure obtained with `GetDisplayOptions`.

It provides this list in the `PaneLayoutOptions` parameter. Each entity contains a list of Area objects defining sizes and positions of panes within a possible layout. This entity defines a full display layout that must be applied “as is” to a video output to switch from one layout to another.

11.3.1 Prerequisites

- A display device providing LayoutOptions
- Receivers that are already configured: see Section 11.4
- IP of the device to configure (represented by the variable *ip*)

11.3.2 Targeted Services and Technologies

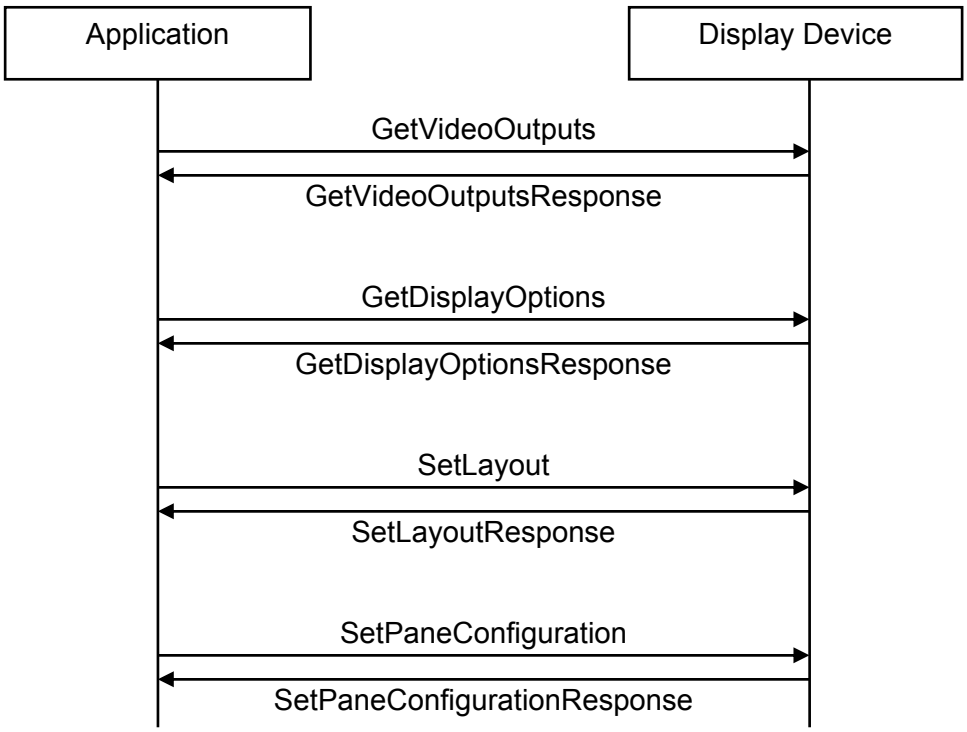
- Device IO service: see [ONVIF/Device-IO-Service] and [deviceio.wsdl]
- Display service: see [ONVIF/Display-Service] and [display.wsdl]

11.3.3 ONVIF::ChangeLayoutByOptions

The general process involves:

1. Getting the current Layout for a video output and the `LayoutOptions` it supports.
2. A corresponding entity of `PaneLayoutOptions` must be used to create a new `PaneLayout` list that replaces or modifies the current Layout before writing back these changes.

The following sample code assumes that currently visible panes shall stay visible. This is achieved by updating the existing Layout entries while maintaining their current relation to the existing `PaneConfigurations`. Further, all panes that are added by the new layout (for example, switching from a 2x2 to a 3x3 layout) get handled by selecting “any” `PaneConfiguration`. Finally, all `PaneConfigurations` that are removed from the current layout are cleaned up by stopping their receiver.



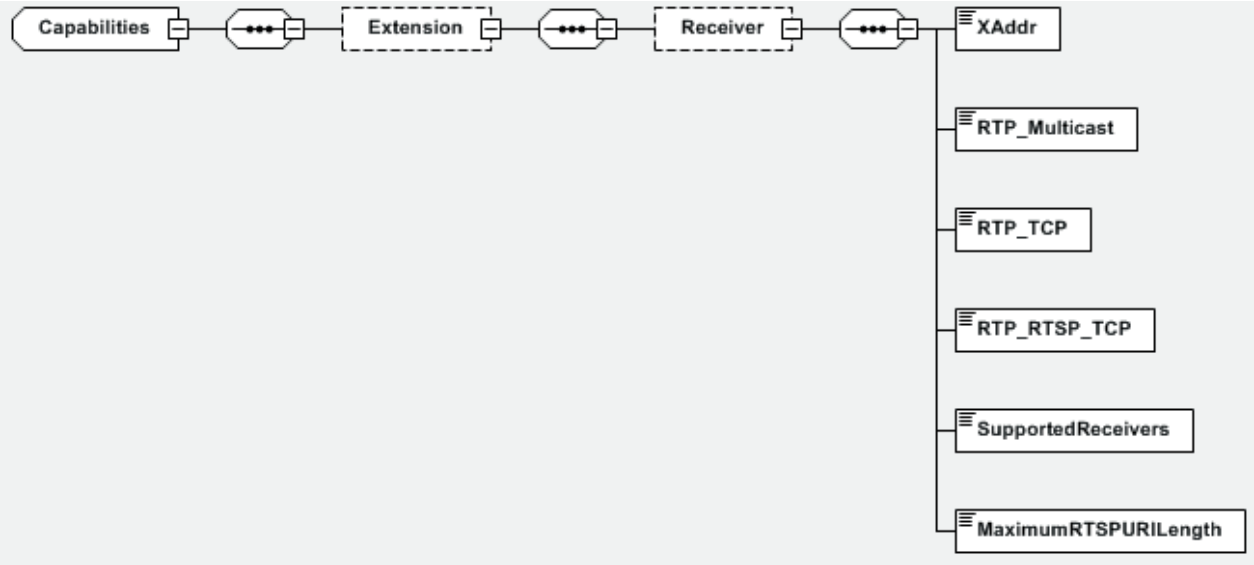

```
// get required services of a display device and select a video output
deviceIOService = getDeviceIOService( ip );
displayService = getDisplayService( ip );

// SOAP trace, see Annex B
videoOutputList = deviceIOService.GetVideoOutputs( );
videoOutput = App.selectVideoOutput( videoOutputList );

layout = videoOutput.Layout;
// now that we have the layout in hands we have to get and apply a new layout option.
// SOAP trace, see Annex B
displayOptionsResponse = displayService.GetDisplayOptions( videoOutput.token );
// we assume the optional parameter is present
paneLayoutOptions = displayOptionsResponse.LayoutOptions.PaneLayoutOptions;
selectedLayoutOption = App.selectNewLayout(paneLayoutOptions );
//wipe out old list of layouts
layout.PaneLayout=PaneLayout()
// now iterate over the areas of the selected LayoutOptions element
// and apply it to the layout
foreach ( Area area in selectedLayoutOption.Area )
{
    PaneLayout paneLayout = PaneLayout();
    paneLayaout.Area.Pane = App.selectPaneConfigurationForNewLayoutArea( index );
    paneLayaout.Area = area;
    App.appendToList( layout.PaneLayout , paneLayout );
}
// now that the PaneLayout list is having content and size of the selected
// PaneLayoutOptions element, we finally write the new layout
// SOAP trace, see Annex B
displayService.SetLayout(videoOutput.token,layout);
```

NOTICE: The visibility of a Pane (`PaneConfiguration`) does not affect the stream state of an associated receiver. If streams that are not visible should be disabled, this must be done explicitly by the application. See [ONVIF/Display-Service] sub chapter 1 covering Panes for details.

TIP: You should shut down and delete streams that are not visible, and delete unused Receivers for `PaneConfigurations` that are not attached to any Layout. This will lower the traffic and free some bandwidth on the LAN.

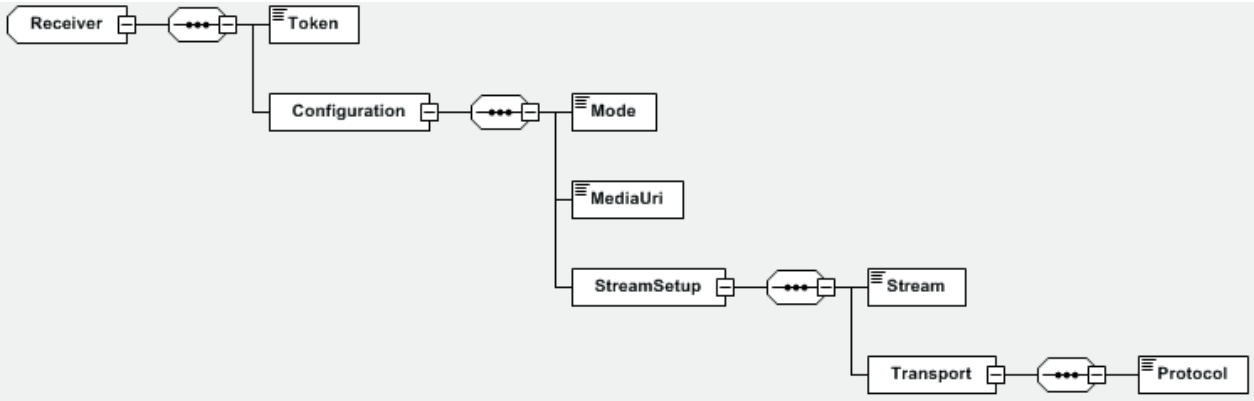
11.4 Configuring a Receiver Based on DecoderCapabilities

This use case describes how to configure a receiver and how to coordinate a display device with a NVT to ensure that the live view can be shown. The first important constraints to consider arise from the capabilities of the receiver. See [ONVIF/Receiver-Configuration].



The most important information that a receiver manages is the transmitter URI. This is limited by the `MaximumRTSPURILength` parameter. Next, `SupportedReceivers` limits the number of receiver instances that a device can manage. The other parameters can be involved in determining the possible transport technologies.

More than one receiver instance might be present, so this example treats it as a single entity.



Nothing unusual is present. Most important is `MediaUri`, which is the reference to the associated transmitter and which is limited by the above mentioned `MaximumRTSPURILength` parameter. `StreamSetup` holds the desired stream configuration, and `Mode` holds the current `Receiver` mode of operation.

The last interesting structure for this use case is `CodingCapabilities`, which declares the decoder capabilities of a pane. The following figure shows the most important structures, considering H.264 streaming and decoding.

Besides the other options for the other video encoding standard, the ONVIF standard provides options for audio encoding and decoding which are not addressed in this document. These ranges are used equally as the corresponding option values of an NVT to select appropriate settings for the NVT encoder.

Because capabilities are covered in previous chapters, this topic is skipped here. For details on how to set up an NVT configuration based on capabilities, see [ONVIF/Display-Service].

11.4.1 Prerequisites

- IP of device to configure (represented by the *ip* variable)
- IP of a NVT to include (represented by the *nvt* variable)
- Token of the video output to associate (represented by the *token* variable)

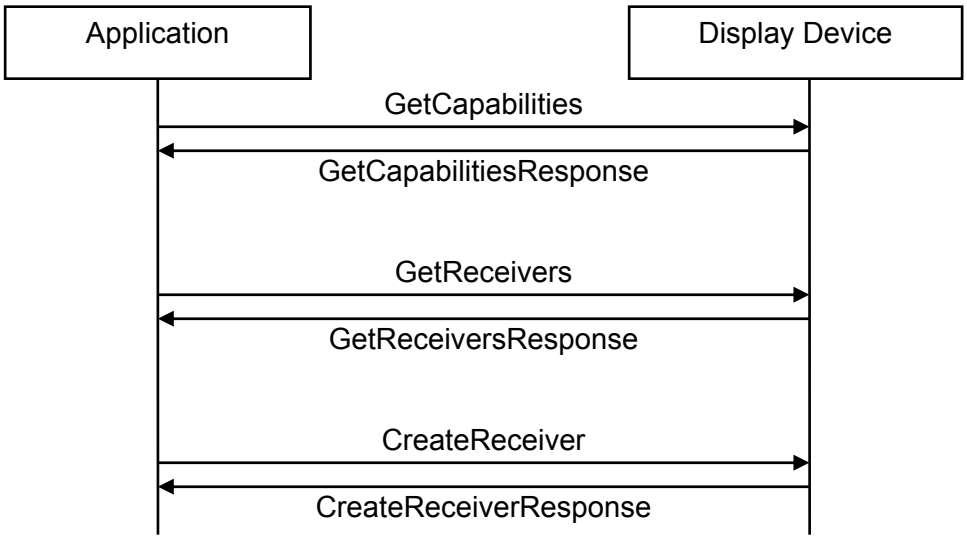
11.4.2 Targeted Services and Technologies

- Device service: see [ONVIF/Device-Service] and [display.wsdl]
- Receiver service: see [ONVIF/Receiver-Configuration] and [receiver.wsdl]

11.4.3 ONVIF::CreateReceiver

In the process shown in the following example:

1. A receiver instance must be selected. In this example, the only save possibility is to create a new receiver which has no connection to any other configuration entity of the according device.
2. `CodingCapabilities` are obtained and applied during NVT configuration, which should provide a `StreamingURI`.
3. These parameters are applied to the selected receiver instance and stored for later use.



```
// get required services of a display device and get the list of current receivers
receiverService = getReceiverService( ip );
deviceService = getDeviceService( ip );

// get the capabilities which provides the maximum number of supported
// receiver instances
capabilities = deviceService.GetCapabilities( );
supportedReceivers = capabilities.Extension.Receiver.SupportedReceivers;

// try to find an unused receiver. SOAP trace, see Annex B
receivers = receiverService.GetReceivers( );
receiver = App.findUnusedReceiver( receivers );

//try to create a new receiver if no unused receiver can be located
if ( ! present(receiver) && size(receivers) < supportedReceivers )
{
    // nice thing about the CreateReceiver interface is that the app doesn't need
    // to take care about creating a unique reference token...
    configuration = new tt::ReceiverConfiguration( );
    configuration.Mode = tt.ReceiverMode( "NeverConnect" );
    configuration.MediaUri = App.configureNvtAndGetStreamUri( nvt , token );
    configuration.StreamSetup.Stream = tt::StreamType( "UDP-Unicast" );
    configuration.StreamSetup.Transport.Protocol = tt::TransportProtocol("HTTP");
    configuration.StreamSetup.Transport.Tunnel = null; //no tunnelling setup
    // SOAP trace, see Annex B
    receiver = receiverService.CreateReceiver(configuration);
}

return receiver;
```

Annex A WSDL-Structures

The ONVIF 2.0 Service Operation Index contains the following 14 ONVIF WSDL schema specifications. These specifications are available in a companion document.

- ONVIF Device Management Service WSDL, version 1.2
- ONVIF Event Service WSDL, version 1.2
- ONVIF Display Service WSDL, version 1.0
- ONVIF Device_IO Service WSDL, version 1.0
- ONVIF Imaging Service WSDL, version 2.0
- ONVIF Media Service WSDL, version 1.2
- ONVIF PTZ Service WSDL, version 2.0
- ONVIF Receiver Service WSDL, version 1.0
- ONVIF Recording_Control Service WSDL, version 1.0
- ONVIF Recording_Search Service WSDL, version 1.0
- ONVIF Remote Discovery Proxy Services WSDL, version 1.1
- ONVIF Replay Service WSDL, version 1.0
- ONVIF Video Analytics Service WSDL, version 2.0
- ONVIF Video Analytics Device_Service WSDL, version 1.0

Annex B SOAP Communication Traces from Use Case Examples

The following SOAP traces are used in the ONVIF use cases described throughout the *Application Programmers Guide*.

B.1 SOAP Communication Trace for Discovery

The following trace refers to Section 4.

In the examples below,

- **Types:** `dn:NetworkVideoTransmitter`
- **Scopes:** `onvif://www.onvif.org/type/video_encoder`
  `onvif://www.onvif.org/type/audio_encoder`
  `onvif://www.onvif.org/hardware/MODEL`
  `onvif://www.onvif.org/name/VENDOR%20MODEL`
  `onvif://www.onvif.org/location/ANY`
- **XAddrs:** `http://169.254.76.145/onvif/services`
  `http://192.168.1.24/onvif/services`
- **Address:** `urn:uuid:a1f48ac2-dc8b-11df-b255-00408c1836b2`

Discovery.Probe message

```
<?xml version="1.0" encoding="UTF-8"?>
<e:Envelope xmlns:e="http://www.w3.org/2003/05/soap-envelope"
 xmlns:w="http://schemas.xmlsoap.org/ws/2004/08/addressing"
 xmlns:d="http://schemas.xmlsoap.org/ws/2005/04/discovery"
 xmlns:dn="http://www.onvif.org/ver10/network/wsdl">
 <e:Header>
  <w:MessageID>uuid:84ede3de-7dec-11d0-c360-f01234567890</w:MessageID>
  <w:To e:mustUnderstand="true">urn:schemas-xmlsoap-org:ws:2005:04:discovery</w:To>
  <w:Action
a:mustUnderstand="true">http://schemas.xmlsoap.org/ws/2005/04/discovery/Pr
obe</w:Action>
 </e:Header>
 <e:Body>
  <d:Probe>
   <d:Types>dn:NetworkVideoTransmitter</d:Types>
  </d:Probe>
 </e:Body>
</e:Envelope>
```

Discovery.ProbeMatch response (one of many similar responses)

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
 xmlns:d="http://schemas.xmlsoap.org/ws/2005/04/discovery"
 xmlns:dn="http://www.onvif.org/ver10/network/wsdl">
  <SOAP-ENV:Header>
    <wsa:MessageID>uuid:84ede3de-e374-11df-b259-00408c1836b2</wsa:MessageID>
    <wsa:RelatesTo>uuid:84ede3de-7dec-11d0-c360-F01234567890</wsa:RelatesTo>
    <wsa:To SOAP-ENV:mustUnderstand="true">http://schemas.xmlsoap.org/ws/2004/08
/addressing/role/anonymous</wsa:To>
    <wsa:Action SOAP-ENV:mustUnderstand="true">http://schemas.xmlsoap.org/ws/200
5/04/discovery/ProbeMatches</wsa:Action>
    <d:AppSequence SOAP-ENV:mustUnderstand="true"
      MessageNumber="3" InstanceId="1287607812"></d:AppSequence>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <d:ProbeMatches>
      <d:ProbeMatch>
        <wsa:EndpointReference>
          <wsa:Address>urn:uuid:a1f48ac2-dc8b-11df-b255-00408c1836b2</wsa:Address>
        </wsa:EndpointReference>
        <d:Types>dn:NetworkVideoTransmitter</d:Types>
        <d:Scopes>onvif://www.onvif.org/type/video_encoder  onvif://www.onvif.org/
type/audio_encoder  onvif://www.onvif.org/hardware/MODEL onvif://www.onvif.org
/name/VENDOR%20MODEL onvif://www.onvif.org/location/ANY</d:Scopes>
        <d:XAddrs>http://169.254.76.145/onvif/services
http://192.168.1.24/onvif/services</d:XAddrs>
        <d:MetadataVersion>1</d:MetadataVersion>
      </d:ProbeMatch>
    </d:ProbeMatches>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.2 SOAP Communication Traces for Initial Setup and Administration

B.2.1 SOAP Communication Traces for First Actions After Discovery

The following traces refer to Section 5.1.

B.2.1.1 GetSystemDateAndTime

Request **device.GetSystemDateAndTime**

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl">
 <SOAP-ENV:Body>
    <tds:GetSystemDateAndTime/>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response to device.GetSystemDateAndTime

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl">
  <SOAP-ENV:Body>
    <tds:GetSystemDateAndTimeResponse>
      <tds:SystemDateAndTime>
        <tt:DateTimeType>NTP</tt:DateTimeType>
        <tt:DaylightSavings>true</tt:DaylightSavings>
        <tt:TimeZone>
          <tt:TZ>CET-1CEST,M3.5.0,M10.5.0</tt:TZ>
        </tt:TimeZone>
        <tt:UTCDateTime>
          <tt:Time>
            <tt:Hour>15</tt:Hour>
            <tt:Minute>52</tt:Minute>
            <tt:Second>25</tt:Second>
          </tt:Time>
          <tt:Date>
            <tt:Year>2010</tt:Year>
            <tt:Month>10</tt:Month>
            <tt:Day>29</tt:Day>
          </tt:Date>
        </tt:UTCDateTime>
        <tt:LocalDateTime>
          <tt:Time>
            <tt:Hour>17</tt:Hour>
            <tt:Minute>52</tt:Minute>
            <tt:Second>25</tt:Second>
          </tt:Time>
          <tt:Date>
            <tt:Year>2010</tt:Year>
            <tt:Month>10</tt:Month>
            <tt:Day>29</tt:Day>
          </tt:Date>
        </tt:LocalDateTime>
      </tds:SystemDateAndTime>
    </tds:GetSystemDateAndTimeResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.2.1.2 GetDeviceInformation

Request device.**GetDeviceInformation**

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl">

<SOAP-ENV:Body>
    <tds:GetDeviceInformation/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl">

<SOAP-ENV:Body>
    <tds:GetDeviceInformationResponse>
      <tds:Manufacturer>VENDOR</tds:Manufacturer>
      <tds:Model>VENDOR MODEL</tds:Model>
      <tds:FirmwareVersion>5.20</tds:FirmwareVersion>
      <tds:SerialNumber>00408C1836B2</tds:SerialNumber>
      <tds:HardwareId>170</tds:HardwareId>
    </tds:GetDeviceInformationResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.2.1.3 GetCapabilities

Request device.**GetCapabilities**

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl">
 <SOAP-ENV:Body>
    <tds:GetCapabilities>
      <tds:Category>All</tds:Category>
    </tds:GetCapabilities>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response to device.GetCapabilities

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:tds="http://www.onvif.org/ver10/device/wsdl">
<SOAP-ENV:Body>
  <tds:GetCapabilitiesResponse>
    <tds:Capabilities>
      <tt:Device>
        <tt:XAddr>http://169.254.76.145/onvif/services</tt:XAddr>
        <tt:Network>
          <tt:IPFilter>true</tt:IPFilter>
          <tt:ZeroConfiguration>true</tt:ZeroConfiguration>
          <tt:IPVersion6>true</tt:IPVersion6>
          <tt:DynDNS>true</tt:DynDNS>
        </tt:Network>
        <tt:System>
          <tt:DiscoveryResolve>true</tt:DiscoveryResolve>
          <tt:DiscoveryBye>true</tt:DiscoveryBye>
          <tt:RemoteDiscovery>false</tt:RemoteDiscovery>
          <tt:SystemBackup>false</tt:SystemBackup>
          <tt:SystemLogging>true</tt:SystemLogging>
          <tt:FirmwareUpgrade>false</tt:FirmwareUpgrade>
          <tt:SupportedVersions>
            <tt:Major>1</tt:Major>
            <tt:Minor>0</tt:Minor>
          </tt:SupportedVersions>
        </tt:System>
        <tt:IO>
          <tt:InputConnectors>1</tt:InputConnectors>
          <tt:RelayOutputs>0</tt:RelayOutputs>
        </tt:IO>
        <tt:Security>
          <tt:TLS1.1>false</tt:TLS1.1>
          <tt:TLS1.2>false</tt:TLS1.2>
          <tt:OnboardKeyGeneration>false</tt:OnboardKeyGeneration>
          <tt:AccessPolicyConfig>false</tt:AccessPolicyConfig>
          <tt:X.509Token>false</tt:X.509Token>
          <tt:SAMLToken>false</tt:SAMLToken>
          <tt:KerberosToken>false</tt:KerberosToken>
          <tt:RELToken>false</tt:RELToken>
        </tt:Security>
      </tt:Device>
      <tt:Events>
        <tt:XAddr>http://169.254.76.145/onvif/services</tt:XAddr>
        <tt:WSSubscriptionPolicySupport>false</tt:WSSubscriptionPolicySupport>
        <tt:WSPullPointSupport>false</tt:WSPullPointSupport>
```

```
        <tt:WSPausableSubscriptionManagerInterfaceSupport>false</tt:WSPausableSubscri
ptionManagerInterfaceSupport>
      </tt:Events>
      <tt:Media>
        <tt:XAddr>http://169.254.76.145/onvif/services</tt:XAddr>
        <tt:StreamingCapabilities>
          <tt:RTPMulticast>true</tt:RTPMulticast>
          <tt:RTP_TCP>true</tt:RTP_TCP>
          <tt:RTP_RTSP_TCP>true</tt:RTP_RTSP_TCP>
        </tt:StreamingCapabilities>
      </tt:Media>
    </tds:Capabilities>
  </tds:GetCapabilitiesResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.2.2 SOAP Communication Traces for Get Network Interface Configuration

The following traces refer to Section 5.2.

B.2.2.1 GetNetworkInterfaces

Request **device.GetNetworkInterfaces**

```
<?xml version='1.0' encoding='utf-8'?>
<soapenv:Envelope
 xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl"
>
  <soapenv:Body>
    <tds:GetNetworkInterfaces/>
  </soapenv:Body>
</soapenv:Envelope>
```

Response – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl">
  <soapenv:Body>
    <tds:GetNetworkInterfacesResponse>
      <tds:NetworkInterfaces token="eth0">
        <tt:Enabled>true</tt:Enabled>
        <tt:Info>
          <tt:Name>eth0</tt:Name>
          <tt:HwAddress> 02:01:23:45:67:89</tt:HwAddress>
          <tt:MTU>1500</tt:MTU>
        </tt:Info>
        <tt:IPv4>
          <tt:Enabled>true</tt:Enabled>
          <tt:Config>
            <tt:Manual>
              <tt:Address>192.168.0.100</tt:Address>
              <tt:PrefixLength>24</tt:PrefixLength>
            </tt:Manual>
            <tt:DHCP>false</tt:DHCP>
          </tt:Config>
        </tt:IPv4>
      </tds:NetworkInterfaces>
    </tds:GetNetworkInterfacesResponse>
```

```
  </soapenv:Body>
</soapenv:Envelope>
```

B.2.3 SOAP Communication Traces for Set Network Interface Configuration

The following traces refer to Section 5.3.

B.2.3.1 SetNetworkInterfaces

Request **device.SetNetworkInterfaces**

```
<?xml version='1.0' encoding='utf-8'?>
<soapenv:Envelope
 xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl"
 xmlns:tt="http://www.onvif.org/ver10/schema">
  <soapenv:Body>
    <tds:SetNetworkInterfaces>
      <tds:InterfaceToken>eth0</tds:InterfaceToken>
      <tds:NetworkInterface>
        <tt:Enabled>true</tt:Enabled>
        <tt:IPv4>
          <tt:Enabled>true</tt:Enabled>
          <tt:Manual>
            <tt:Address>192.168.0.200</tt:Address>
            <tt:PrefixLength>24</tt:PrefixLength>
          </tt:Manual>
          <tt:DHCP>false</tt:DHCP>
        </tt:IPv4>
      </tds:NetworkInterface>
    </tds:SetNetworkInterfaces>
  </soapenv:Body>
</soapenv:Envelope>
```

Response - on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl">
  <soapenv:Body>
    <tds:SetNetworkInterfacesResponse>
      <tt:RebootNeeded>true</tt:RebootNeeded>
    </tds:SetNetworkInterfacesResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

B.2.3.2 SystemReboot

Request **device.SystemReboot**

```
<?xml version='1.0' encoding='utf-8'?>
<soapenv:Envelope
 xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl">
  <soapenv:Body>
    <tds:SystemReboot/>
  </soapenv:Body>
</soapenv:Envelope>
```

Response – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl">
  <soapenv:Body>
    <tds:SystemRebootResponse>
      <tt:Message>Rebooting in 30 seconds.</tt:Message>
    </tds:SystemRebootResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

B.2.4 SOAP Communication Traces for Time synchronization Including NTP Configuration (Set Manually)

The following traces refer to Section 5.4.

B.2.4.1 SetNTP

Request **device.SetNTP**

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl">
  <soapenv:Body>
    <tds:SetNTP>
      <tds:FromDHCP>false</tds:FromDHCP>
      <tds:NTPManual>
        <tt:Type>IPv4</tt:Type>
        <tt:IPv4Address>192.168.10.1</tt:IPv4Address>
      </tds:NTPManual>
    </tds:SetNTP>
  </soapenv:Body>
</soapenv:Envelope>
```

Response – on success

```
<?xml version='1.0' encoding='utf-8'?>
<soapenv:Envelope
 xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl">
  <soapenv:Body>
    <tds:SetNTPResponse/>
  </soapenv:Body>
</soapenv:Envelope>
```

B.2.4.2 SetSystemDateAndTime

Request **device.SetSystemDateAndTime**

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl">
  <soapenv:Body>
    <tds:SetSystemDateAndTime>
      <tt:DateTimeType>NTP</tt:DateTimeType>
      <tt:DaylightSavings>false</tt:DaylightSavings>
    </tds:SetSystemDateAndTime>
  </soapenv:Body>
</soapenv:Envelope>
```

Response – on success

```
<?xml version='1.0' encoding='utf-8'?>
<soapenv:Envelope
 xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl">
  <soapenv:Body>
    <tds:SetSystemDateAndTimeResponse/>
  </soapenv:Body>
</soapenv:Envelope>
```

B.2.5 SOAP Communication Traces for Time synchronization Including NTP Configuration

The following traces refer to Section 5.5.

B.2.5.1 SetNTP

Reqest **device.SetNTP**

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl">
  <soapenv:Body>
    <tds:SetNTP>
      <tds:FromDHCP>true</tds:FromDHCP>
    </tds:SetNTP>
  </soapenv:Body>
</soapenv:Envelope>
```

For the response, see the previous section.

B.2.5.2 SetSystemDateAndTime

Request **device.SetSystemDateAndTime**

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:tds="http://www.onvif.org/ver10/device/wsdl">
  <soapenv:Body>
    <tds:SetSystemDateAndTime>
      <tt:DateTimeType>NTP</tt:DateTimeType>
      <tt:DaylightSavings>false</tt:DaylightSavings>
    </tds:SetSystemDateAndTime>
  </soapenv:Body>
</soapenv:Envelope>
```

For the response, see previous section.

B.2.6 SOAP Communication Traces for Backup System Configuration Files from a Device

The following traces refer to Section 5.6.

B.2.6.1 device.GetSystemBackup

Request **GetSystemBackup**

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
  <SOAP-ENV:Body>
    <tds:GetSystemBackup xmlns="http://www.onvif.org/ver10/device/wsdl"/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
    <SOAP-ENV:Body>
        <tds:GetSystemBackupResponse xmlns="http://www.onvif.org/ver10/device/wsdl">
            <BackupFiles>
                <Name xmlns="http://www.onvif.org/ver10/schema">DeviceX-
00:04:7D:01:EF:40-backup</Name>
                <Data xmlns="http://www.onvif.org/ver10/schema"><xop:Include
href="cid:1.6333358458759375000@example.org"/>
                </Data>
            </BackupFiles>
        </ tds:GetSystemBackupResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.2.6.2 HTTP / MTOM Communication Trace

HTTP / MTOM Request trace

```
POST /onvif/services HTTP/1.1
Connection: close
Content-Type: application/soap+xml;charset=UTF-
8;action="http://www.onvif.org/ver10/device/wsdl/GetSystemBackup"
User-Agent: XXX
Host: 10.220.233.73
Content-Length: 216
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsdl="http://www.onvif.org/ver10/device/wsdl">

<soap:Body>
      <wsdl:GetSystemBackup/>
   </soap:Body>
</soap:Envelope>
```

HTTP / MTOM Response trace

```
HTTP/1.1 200 OK
Server: XXX
Content-Type: multipart/related;
boundary="==LDfy0hnQoWbx8sVSPKwmi4+pH6kCThEekjRYF4otODfKw0c/+MlhEjKLduOx==";
type="application/xop+xml"; start="<SOAP-ENV:Envelope>"; start-
info="application/soap+xml; charset=utf-8"
Content-Length: 51917
Connection: close
Date: Mon, 13 Dec 2010 23:39:59 GMT
--==LDfy0hnQoWbx8sVSPKwmi4+pH6kCThEekjRYF4otODfKw0c/+MlhEjKLduOx==
Content-Type: application/xop+xml; charset=utf-8; type=application/soap+xml
Content-Transfer-Encoding: binary
Content-ID:  1.633335845875937500@example.org
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:SOAP-ENC="http://www.w3.org/2003/05/soap-encoding"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tds="http://www.onvif.org/ver10/device/wsdl"
xmlns:tt="http://www.onvif.org/ver10/schema">
<SOAP-ENV:Body>
<tds:GetSystemBackupResponse>
<tds:BackupFiles>
<tt:Name>DeviceX-00:04:7D:01:EF:40-backup</tt:Name>
<tt:Data xmime5:contentType="application/octet-stream">
<xop:Include href="cid:id3"/>
</tt:Data>
</ tds:BackupFiles>
</ tds:GetSystemBackupResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
--==LDfy0hnQoWbx8sVSPKwmi4+pH6kCThEekjRYF4otODfKw0c/+MlhEjKLduOx==
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
Content-ID: <id3>
...
```

B.2.7 SOAP Communication Traces for Restore System Configuration Files to a Device

The following traces refer to Section 5.7.

B.2.7.1 device.RestoreSystem

Request

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
  <SOAP-ENV:Body>
    <tds:RestoreSystem xmlns="http://www.onvif.org/ver10/device/wsdl">
      <BackupFiles>
        <Name xmlns="http://www.onvif.org/ver10/schema">STRING</Name>
<Data xmlns="http://www.onvif.org/ver10/schema"><xop:Include
href="cid:1.633335845875937500@example.org"/>
                </Data>
      </BackupFiles>
    </ tds:RestoreSystem>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response - on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
  <SOAP-ENV:Body>
    <tds:RestoreSystemResponse xmlns="http://www.onvif.org/ver10/device/wsdl"/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response – on failure, when the content of section BackupFiles was wrong

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <SOAP-ENV:Code>
        <SOAP-ENV:Value>env:Sender</SOAP-ENV:Value>
        <SOAP-ENV:Subcode>
          <SOAP-ENV:Value>ter:InvalidArgVal</SOAP-ENV:Value>
          <SOAP-ENV:Subcode>
            <SOAP-ENV:Value>ter:InvalidBackupFile</SOAP-ENV:Subcode>
          </SOAP-ENV:Subcode>
        </SOAP-ENV:Subcode>
      </SOAP-ENV:Code>
      <SOAP-ENV:Reason>
        <SOAP-ENV:Text xml:lang="en">The backup file(s) are invalid.</SOAP-ENV:Text>
      </SOAP-ENV:Reason>
     </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

For the HTTP communication trace for MTOM, see the example trace listed in Section 5.6.3.

B.2.8 SOAP Communication Traces for Start System Restore via HTTP Post

The following traces refer to Section 5.8.

B.2.8.1 device.StartSystemRestore

Request StartSystemRestore

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsdl="http://www.onvif.org/ver10/device/wsdl">
   <SOAP-ENV:Body>
      <tds:StartSystemRestore xmlns="http://www.onvif.org/ver10/device/wsdl"/>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsdl="http://www.onvif.org/ver10/device/wsdl">
  <SOAP-ENV:Body>
    <tds:StartSystemRestoreResponse xmlns="http://www.onvif.org/ver10/device/wsdl"/>
      <UploadUri xmlns="http://www.onvif.org/ver10/schema">http://SOMEURI</UploadUri>
      <ExpectedDownTime
xmlns="http://www.onvif.org/ver10/schema">30</ExpectedDownTime>
    </ tds:StartSystemRestoreResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.2.8.2 HTTP Communication Traces ('Unsupported Media Type', ‘uploaded file was invalid')

Request	Response
POST http://UploadURI HTTP/1.0 Content-Type: application/octet-stream	HTTP/1.1 415

B.2.8.3 HTTP Communication Trace ('Internal Server Error', ‘error at the device')

Request	Response
POST http://UploadURI HTTP/1.0 Content-Type: application/octet-stream	HTTP/1.1 500

B.2.8.4 HTTP Communication Trace (‘OK’, ‘restore successful’)

Request	Response
POST http://UploadURI HTTP/1.0 Content-Type: application/octet-stream	HTTP/1.1 200 OK

B.3 SOAP Communication Traces for Security

B.3.1 SOAP Communication Trace for Validating WS-UsernameToken

The following trace refers to Section 6.1.2.

B.3.1.1 GetUsers

REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:tt="http://www.onvif.org/ver10/schema">
  <soap12:Body>
    <GetUsers xmlns="http://www.onvif.org/ver10/device/wsdl"/>
  </soap12:Body>
</soap12:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tt="http://www.onvif.org/ver10/schema">
  <soap12:Body>
    <GetUsersResponse xmlns="http://www.onvif.org/ver10/device/wsdl">
      <User>
        <tt:Username>admin</tt:Username>
        <tt:UserLevel>Administrator</tt:UserLevel>
      </User>
    </GetUsersResponse>
  </soap12:Body>
</soap12:Envelope>
```

B.3.2 SOAP Communication Trace for User Management

B.3.2.1 Registering the User

The following trace refers to Section 6.2.1.

B.3.2.1.1 CreateUsers

REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tt="http://www.onvif.org/ver10/schema">
  <soap12:Body>
    <CreateUsers xmlns="http://www.onvif.org/ver10/device/wsdl">
      <User>
        <tt:Username>newusername</tt:Username>
        <tt:Password>newuserpassword</tt:Password>
        <tt:UserLevel>Administrator</tt:UserLevel>
      </User>
    </CreateUsers>
  </soap12:Body>
</soap12:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
  <soap12:Body>
    <CreateUsersResponse xmlns="http://www.onvif.org/ver10/device/wsdl">
    </CreateUsersResponse>
  </soap12:Body>
</soap12:Envelope>
```

B.3.2.2 Changing the Password

The following trace refers to Section 6.2.1.

B.3.2.2.1 SetUser

REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tt="http://www.onvif.org/ver10/schema">
  <soap12:Body>
    <SetUser xmlns="http://www.onvif.org/ver10/device/wsdl">
      <User>
        <tt:Username>username</tt:Username>
        <tt:Password>newpassword</tt:Password>
        <tt:UserLevel>Administrator</tt:UserLevel>
      </User>
    </SetUser>
  </soap12:Body>
</soap12:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:enc="http://www.w3.org/2003/05/soap-encoding"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <env:Body>
    <SetUserResponse xmlns="http://www.onvif.org/ver10/device/wsdl">
    </SetUserResponse>
  </env:Body>
</env:Envelope>
```

B.3.2.3 Deleting the User

The following trace refers to Section 6.2.3.

B.3.2.3.1 DeleteUsers

REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap12:Body>
    <DeleteUsers xmlns="http://www.onvif.org/ver10/device/wsdl">
      <Username>deleteusername</Username>
    </DeleteUsers>
  </soap12:Body>
</soap12:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:enc="http://www.w3.org/2003/05/soap-encoding"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <env:Body>
    <DeleteUsersResponse xmlns="http://www.onvif.org/ver10/device/wsdl">
    </DeleteUsersResponse>
  </env:Body>
</env:Envelope>
```

B.3.3 SOAP Communication Traces for Certificate Management and Usage

B.3.3.1 Setting Up a Self-Signed Certificate of the Device

The following traces refer to Section 6.3.1.

B.3.3.1.1 CreateCertificate

The Subject field in this example is vendor-specific and has been omitted. For support on how to create the subject for the specific request, follow up with the appropriate vendor.

REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tt="http://www.onvif.org/ver10/schema" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
  <soap12:Header>
    <wsse:Security>
      <wsu:Timestamp wsu:Id="Time">
        <wsu:Created>2010-12-15T09:44:53Z</wsu:Created>
        <wsu:Expires>2010-12-15T09:45:03Z</wsu:Expires>
      </wsu:Timestamp>
      <wsse:UsernameToken wsu:Id="User">
        <wsse:Username>admin</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-
```

```
1.0#PasswordDigest">aUn1yvwgh/rm4a/srO8hboMT6ms=</wsse:Password>
        <wsse:Nonce>AE0aDtVhZk6N/VBChMyiCw==</wsse:Nonce>
        <wsu:Created>2010-12-15T09:44:53Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap12:Header>
  <soap12:Body>
    <CreateCertificate xmlns="http://www.onvif.org/ver10/device/wsdl">
      <CertificateID>SelfSigned1</CertificateID>
      <Subject>
        <!-- Vendor specific parameter -->
      </Subject>
      <ValidNotAfter>2020-10-01T09:00:00</ValidNotAfter>
    </CreateCertificate>
  </soap12:Body>
</soap12:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:enc="http://www.w3.org/2003/05/soap-encoding"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
xmlns:xop="http://www.w3.org/2004/08/xop/include"
xmlns:tt="http://www.onvif.org/ver10/schema">
  <env:Body>
    <CreateCertificateResponse xmlns="http://www.onvif.org/ver10/device/wsdl">
      <NvtCertificate>
        <tt:CertificateID>SelfSigned1</tt:CertificateID>
        <tt:Certificate>
          <tt:Data>MIICXzCCAcigAwIBAgIGAIBFWaM7MA0GCSqGSIb3DQEBBQUAMHMxCzAJBgNVBAYTAkp
          QMQ4wDAYDVQQIEwVUb2t5bzEPMA0GA1UEBxMGTWVndXJvMRcwFQYDVQQKEw4gR3JlZW4gTGltaXR
          lZDEaMBgGA1UECxMRVGVjaG5vbG9neSBDZW50ZXIxDjAMBgNVBAMTBWhvc3QxMB4XDTAxMDEwMTA
          wMDAwMFoXDTIwMTAwMTA5MDAwMFowczELMAkGA1UEBhMCSlAxDjAMBgNVBAgTBVRva3lvMQ8wDQY
          DVQQHEwZNZWd1cm8xFzAVBgNVBAoTDiBHcmVlbiBMaW1pdGVkMRowGAYDVQQLExFUZWNobm9sb2d
          5IENlbnRlcjEOMAwGA1UEAxMFaG9zdDEwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMcwKkT
          3m2vufWgOX73iLnK6O0pUnQ08QZRcpjBXXXuOZKVkrXRYsU4V65F+0BK6QPZcAZcnEQPrIkhbXfh
          QKmeHMTpXV+W6/nHovv3qSA9oVxcsNumI6FFiSYnnuyGsHyvu2wiVE24y+uzTEQrQPi0/VvrqE16
          cw3EJd09ogu+TAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAJyr6A8q91pDWqm27fvgyUYXE1Mhz17o
          JLvNS/HRg6wrHxku8UdAYPI6RchqVGHl/KeTZbYJfYCB/2D8Ltbh1sJmJ750Ex6FvZJC7y45vc11
          oSKoPoaTwj/AZ+1MHK06p6NKqod3rYo+xEk25hedwUXtFpAvb0qgWRekrsYKjhR0=</tt:Data>
        </tt:Certificate>
      </NvtCertificate>
    </CreateCertificateResponse>
  </env:Body>
</env:Envelope>
```

B.3.3.1.2 GetCertificatesStatus

REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tt="http://www.onvif.org/ver10/schema" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
```

```
  <soap12:Header>
    <wsse:Security>
      <wsu:Timestamp wsu:Id="Time">
        <wsu:Created>2010-12-15T09:52:27Z</wsu:Created>
        <wsu:Expires>2010-12-15T09:52:37Z</wsu:Expires>
      </wsu:Timestamp>
      <wsse:UsernameToken wsu:Id="User">
        <wsse:Username>admin</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-
1.0#PasswordDigest">zt64Tr8gJbjgFhx3uWPQOl0vlCk=</wsse:Password>
        <wsse:Nonce>VmJLgRgKkk6bk7dqMWadEQ==</wsse:Nonce>
        <wsu:Created>2010-12-15T09:52:27Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap12:Header>
  <soap12:Body>
    <GetCertificatesStatus xmlns="http://www.onvif.org/ver10/device/wsdl" />
  </soap12:Body>
</soap12:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:enc="http://www.w3.org/2003/05/soap-encoding"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
xmlns:xop="http://www.w3.org/2004/08/xop/include"
xmlns:tt="http://www.onvif.org/ver10/schema">
  <env:Body>
    <GetCertificatesStatusResponse xmlns="http://www.onvif.org/ver10/device/wsdl">
      <CertificateStatus>
        <tt:CertificateID>SelfSigned1</tt:CertificateID>
        <tt:Status>true</tt:Status>
      </CertificateStatus>
    </GetCertificatesStatusResponse>
  </env:Body>
</env:Envelope>
```

B.3.3.1.3 SetCertificatesStatus

REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tt="http://www.onvif.org/ver10/schema" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
  <soap12:Header>
    <wsse:Security>
      <wsu:Timestamp wsu:Id="Time">
        <wsu:Created>2010-12-15T09:58:36Z</wsu:Created>
        <wsu:Expires>2010-12-15T09:58:46Z</wsu:Expires>
      </wsu:Timestamp>
      <wsse:UsernameToken wsu:Id="User">
        <wsse:Username>admin</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-
```

```
1.0#PasswordDigest">I7sE3LvaAHTyhUPMmHgxrxTmM8E=</wsse:Password>
        <wsse:Nonce>iQySVaOOjE2v+dPWzkUuXg==</wsse:Nonce>
        <wsu:Created>2010-12-15T09:58:36Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap12:Header>
  <soap12:Body>
    <SetCertificatesStatus xmlns="http://www.onvif.org/ver10/device/wsdl">
      <CertificateStatus>
        <tt:CertificateID>SelfSigned1</tt:CertificateID>
        <tt:Status>true</tt:Status>
      </CertificateStatus>
    </SetCertificatesStatus>
  </soap12:Body>
</soap12:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:enc="http://www.w3.org/2003/05/soap-encoding"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
xmlns:xop="http://www.w3.org/2004/08/xop/include"
xmlns:tt="http://www.onvif.org/ver10/schema">
  <env:Body>
    <SetCertificatesStatusResponse xmlns="http://www.onvif.org/ver10/device/wsdl">
    </SetCertificatesStatusResponse>
  </env:Body>
</env:Envelope>
```

B.3.3.2 Getting a PKCS #10 Certificate Signature Request from the Device

The following trace refers to Section 6.3.2.

B.3.3.2.1 GetPkcs10Request

The subject field in this example is vendor-specific and has been omitted. For support on how to create the subject for the specific request, follow up with the appropriate vendor.

REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tt="http://www.onvif.org/ver10/schema" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
  <soap12:Header>
    <wsse:Security>
      <wsu:Timestamp wsu:Id="Time">
        <wsu:Created>2010-12-15T10:11:06Z</wsu:Created>
        <wsu:Expires>2010-12-15T10:11:16Z</wsu:Expires>
      </wsu:Timestamp>
      <wsse:UsernameToken wsu:Id="User">
        <wsse:Username>admin</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-
1.0#PasswordDigest">6QHS1lyd/hkTY+0Px/LonCtwJs0=</wsse:Password>
```

```
        <wsse:Nonce>G2bfEGaQ/kivDYt2pWk/Lw==</wsse:Nonce>
        <wsu:Created>2010-12-15T10:11:06Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap12:Header>
  <soap12:Body>
    <GetPkcs10Request xmlns="http://www.onvif.org/ver10/device/wsdl">
      <CertificateID>SelfSigned2</CertificateID>
      <Subject>
        <!-- Vendor specific parameter -->
      </Subject>
    </GetPkcs10Request>
  </soap12:Body>
</soap12:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:enc="http://www.w3.org/2003/05/soap-encoding"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
xmlns:xop="http://www.w3.org/2004/08/xop/include"
xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:xm="http://www.w3.org/2005/05/xmlmime">
  <env:Body>
    <GetPkcs10RequestResponse xmlns="http://www.onvif.org/ver10/device/wsdl">
      <Pkcs10Request>
        <tt:Data>LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSBSRVFVRVNULS0tLS0KTUlJQnNqQ0NBUnNDQVFBd
        2NqRUxNQWtHQTFVRUJoTUNTbEF4RGpBTUJnTlZCQWdUQlZSdmEzbHZNUTh3RFFZRApWUVFIRXdaTlp
        XZDFjbTh4RmpBVUJnTlZCQW9URFVkeVpXVnVJRXhwYldsMFpXUXhHakFZQmdOVkJBc1RFVlJsClkya
        HViMnh2WjNrZ1EyVnVkR1Z5TVE0d0RBWURWUVFERXdWb2IzTjBNVENCbnpBTkJna3Foa2lHOXcwQkF
        RRUYKQUFPQmpRQXdnWWtDZ1lFQXh6QXFSUGViYSs1OWFBNWZ2ZUl1Y3JvN1NsU2REVHhCbEZ5bU1GZ
        GRlNDVrcFdTdApkRml4VGhYcmtYN1FFcnBBOWx3Qmx5Y1JBK3NpU0Z0ZCtGQXFaNGN4T2xkWDVicit
        jZWkrL2VwSUQyaFhGeXcyCjZZam9VV0pKaWVlN0lhd2ZLKzdiQ0pVVGJqTDY3Tk1SQ3RBK0xUOVcrd
        W9UWHB6RGNRbDNUMmlDNzVNQ0F3RUEKQWFBQU1BMEdDU3FHU0liM0RRRUJCUVVBQTRHQkFMZjBEaGd
        3QlhjVXBkMjV5SDJUd09mbnJ0cUZpWlA4ekxFbQpxRzU5TDdhdkZEbVFxcmdMWU1qQkpkc21OVitmN
        3VoQ2doM2diNWVDQmNnR2wwWCtuNFhOSEw2M3dVQm9vTTJoCmRUeXNHSUpJM1hDYUx5SS90eEl6OUh
        Mc01NS2Z0VVRacXZjd0x2QWpWK1lFYldvV2FtNzlBWnc3cTFxeWNnZnEKNFRRVmFHR04KLS0tLS1FT
        kQgQ0VSVElGSUNBVEUgUkVRVUVTVC0tLS0tCgA=</tt:Data>
      </Pkcs10Request>
    </GetPkcs10RequestResponse>
  </env:Body>
</env:Envelope>
```

B.3.3.3 Setting Up a Signed Certificate of the Device (Except for a Self-Signed Certificate)

The following trace refers to Section 6.3.3.

B.3.3.3.1 LoadCertificates

REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:tt="http://www.onvif.org/ver10/schema" xmlns:wsse="http://docs.oasis-
       open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
       xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
```

```
       utility-1.0.xsd">
  <soap12:Header>
    <wsse:Security>
      <wsu:Timestamp wsu:Id="Time">
        <wsu:Created>2010-12-15T10:31:07Z</wsu:Created>
        <wsu:Expires>2010-12-15T10:31:17Z</wsu:Expires>
      </wsu:Timestamp>
      <wsse:UsernameToken wsu:Id="User">
        <wsse:Username>admin</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
       username-token-profile-
       1.0#PasswordDigest">NWSuWYhaswddfESWz3p1EFqOVkU=</wsse:Password>
        <wsse:Nonce>0IVhwRSiWkacTRUs3JWm6w==</wsse:Nonce>
        <wsu:Created>2010-12-15T10:31:07Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap12:Header>
  <soap12:Body>
    <LoadCertificates xmlns="http://www.onvif.org/ver10/device/wsdl">
      <NVTCertificate>
        <tt:CertificateID>CASigned1</tt:CertificateID>
        <tt:Certificate>
          <tt:Data>MIIEajCCA9OgAwIBAgIQbypes1ZWrRNc6WboYpTs0TANBgkqhkiG9w0BAQUFADCB5zE
          LMAkGA1UEBhMCVVMxFzAVBgNVBAoTDlZlcmlTaWduLCBJbmMuMR8wHQYDVQQLExZGT1IgVEVTVCB
          QVVJQT1NFUyBPTkxZMR8wHQYDVQQLExZWZXJpU2lnbiBUcnVzdCBOZXR3b3JrMUMwQQYDVQQLEzp
          UZXJtcyBvZiB1c2UgYXQgaHR0cHM6Ly93d3cudmVyaXNpZ24uY29tL2Nwcy90ZXN0Y2EvIChjKTA
          3MTgwNgYDVQQDEy9WZXJpU2lnbiBDbGFzcyAzIFNlY3VyZSBTZXJ2ZXIgMTAyNC1iaXQgVGVzdCB
          DQTAeFw0xMDA2MjIwMDAwMDBaFw0xMDA3MDYyMzU5NTlaMIGjMQswCQYDVQQGEwJKUDERMA8GA1U
          ECBMIS2FtYWdhd2ExETAPBgNVBAcUCFlva29oYW1hMRIwEAYDVQQKFAlQYW5hc29uaWMxDTALBgN
          VBAsUBFBGREMxOjA4BgNVBAsUMVRlcm1zIG9mIHVzZSBhdCB3d3cudmVyaXNpZ24uY29tL2Nwcy9
          0ZXN0Y2EgKGMpMDUxDzANBgNVBAMUBmNhbWVyYTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYE
          AtclG+P9Uzj1C0y3y9Hk6jJkgnrjLbCfpsNhOYPIiR/OJQntvREpcw8ktvKjkVKiG7K5MnZVoDdi
          KmcYLf5PMVljFnw/dtUEalQXyYK1K6Wpv3pQFNP+G9903Y+w3lYGTUBPn2YuEouJjwfveLTYessC
          u21O6jo6Mo3UGgUy5kX0CAwEAAaOCAVcwggFTMAkGA1UdEwQCMAAwCwYDVR0PBAQDAgWgMD0GA1U
          dHwQ2MDQwMqAwoC6GLGh0dHA6Ly9jcmwudmVyaXNpZ24uY29tL1NWUjEwMjRUcmlhbDIwMDcuY3J
          sMEoGA1UdIARDMEEwPwYKYIZIAYb4RQEHFTAxMC8GCCsGAQUFBwIBFiNodHRwczovL3d3dy52ZXJ
          pc2lnbi5jb20vY3BzL3Rlc3RjYTAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIwHwYDVR0
          jBBgwFoAU6XX2ekWwDNKb42eN0kQT1JHEfXYwbgYIKwYBBQUHAQwEYjBgoV6gXDBaMFgwVhYJaW1
          hZ2UvZ2lmMCEwHzAHBgUrDgMCGgQUS2u5KJYGDLvQUjibKaxLB4shBRgwJhYkaHR0cDovL2xvZ28
          udmVyaXNpZ24uY29tL3ZzbG9nbzEuZ2lmMA0GCSqGSIb3DQEBBQUAA4GBALdm+PMsUq2zTSpDsUL
          aVZtQhyI0P3guVINkypPyxPCKb8MKCHLI8DmEjWeZe8oohJlvX8pyvlIdXzdpqXrFsy+EkgSoykF
          GR/EnYN9uZ8HuUNPQqyKy9248FEAFnPheffdXDosFS6jtJIfYaIe6YKQr4WTNWIgCt3h8c+B5t7x
          A</tt:Data>
        </tt:Certificate>
      </NVTCertificate>
    </LoadCertificates>
  </soap12:Body>
</soap12:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:enc="http://www.w3.org/2003/05/soap-encoding"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tt="http://www.onvif.org/ver10/schema">
  <env:Body>
    <LoadCertificatesResponse xmlns="http://www.onvif.org/ver10/device/wsdl">
    </LoadCertificatesResponse>
  </env:Body>
</env:Envelope>
```

B.3.3.4 Getting Information About Device Certificates

The following trace refers to Section 6.3.4.

B.3.3.4.1 GetCertificateInformation

REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:tt="http://www.onvif.org/ver10/schema" xmlns:wsse="http://docs.oasis-
       open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
       xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
       utility-1.0.xsd">
  <soap12:Header>
    <wsse:Security>
      <wsu:Timestamp wsu:Id="Time">
        <wsu:Created>2010-12-15T10:31:07Z</wsu:Created>
        <wsu:Expires>2010-12-15T10:31:17Z</wsu:Expires>
      </wsu:Timestamp>
      <wsse:UsernameToken wsu:Id="User">
        <wsse:Username>admin</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
        username-token-profile-
        1.0#PasswordDigest">NWSuWYhaswddfESWz3p1EFqOVkU=</wsse:Password>
        <wsse:Nonce>0IVhwRSiWkacTRUs3JWm6w==</wsse:Nonce>
        <wsu:Created>2010-12-15T10:31:07Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap12:Header>
  <soap12:Body>
    <GetCertificateInformation xmlns="http://www.onvif.org/ver10/device/wsdl">
      <CertificateID>CASigned1</CertificateID>
    </GetCertificateInformation>
  </soap12:Body>
</soap12:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:tt="http://www.onvif.org/ver10/schema">
  <soap12:Body>
    <GetCertificateInformationResponse xmlns="http://www.onvif.org/ver10/device/wsdl">
      <CertificateInformation>
        <tt:CertificateID>CASigned1<tt:CertificateID/>
        <tt:IssuerDN>CN=ABC CA;OU=ABC Network;O=ABC;C=US</tt:IssuerDN>
        <tt:SubjectDN>CN=host1;OU=Technology Center;O=Green
        Limited;L=Meguro;ST=Tokyo;C=JP</tt:SubjectDN>
        <tt:KeyLength>1024</tt:KeyLength>
        <tt:Version>3</tt:Version>
        <tt:SerialNum>00000000</tt:SerialNum>
        <tt:SignatureAlgorithm>sha1RSA</tt:SignatureAlgorithm>
        <tt:Validity>
          <tt:From>2010-12-17T00:00:00Z</tt:From>
          <tt:Until>2020-12-17T00:00:00Z</tt:Until>
        </tt:Validity>
      </CertificateInformation>
    </GetCertificateInformationResponse>
  </soap12:Body>
```

```
</soap12:Envelope>
```

B.3.3.5 Deleting the Certificates of a Device

The following trace refers to Section 6.3.5.

B.3.3.5.1 DeleteCertificates

REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:tt="http://www.onvif.org/ver10/schema" xmlns:wsse="http://docs.oasis-
       open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
       xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
       utility-1.0.xsd">
  <soap12:Header>
    <wsse:Security>
      <wsu:Timestamp wsu:Id="Time">
        <wsu:Created>2010-12-16T04:14:54Z</wsu:Created>
        <wsu:Expires>2010-12-16T04:15:04Z</wsu:Expires>
      </wsu:Timestamp>
      <wsse:UsernameToken wsu:Id="User">
        <wsse:Username>admin</wsse:Username>
        <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
        username-token-profile-
        1.0#PasswordDigest">6pm+oKs7OY6DcLRCJBgAGZQ4WfA=</wsse:Password>
        <wsse:Nonce>7yEbAncRKUuT8p3X+9T69A==</wsse:Nonce>
        <wsu:Created>2010-12-16T04:14:54Z</wsu:Created>
      </wsse:UsernameToken>
    </wsse:Security>
  </soap12:Header>
  <soap12:Body>
    <DeleteCertificates xmlns="http://www.onvif.org/ver10/device/wsdl">
      <CertificateID>SelfSigned1</CertificateID>
    </DeleteCertificates>
  </soap12:Body>
</soap12:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
       xmlns:enc="http://www.w3.org/2003/05/soap-encoding"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
       xmlns:xop="http://www.w3.org/2004/08/xop/include">
  <env:Body>
    <DeleteCertificatesResponse xmlns="http://www.onvif.org/ver10/device/wsdl">
    </DeleteCertificatesResponse>
  </env:Body>
</env:Envelope>
```

B.3.4 Real-Time Streaming via RTP / RTSP / HTTPS

The following traces refer to Section 6.4.3.

B.3.4.1 GetNetworkProtocols

REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:tt="http://www.onvif.org/ver10/schema">
  <soap12:Body>
    <GetNetworkProtocols xmlns="http://www.onvif.org/ver10/device/wsdl" />
  </soap12:Body>
</soap12:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
       xmlns:enc="http://www.w3.org/2003/05/soap-encoding"
       xmlns:tt="http://www.onvif.org/ver10/schema">
  <env:Body>
    <GetNetworkProtocolsResponse xmlns="http://www.onvif.org/ver10/device/wsdl">
      <NetworkProtocols>
        <tt:Name>HTTP</tt:Name>
        <tt:Enabled>true</tt:Enabled>
        <tt:Port>80</tt:Port>
      </NetworkProtocols>
      <NetworkProtocols>
        <tt:Name>RTSP</tt:Name>
        <tt:Enabled>true</tt:Enabled>
        <tt:Port>554</tt:Port>
      </NetworkProtocols>
      <NetworkProtocols>
        <tt:Name>HTTPS</tt:Name>
        <tt:Enabled>false</tt:Enabled>
        <tt:Port>443</tt:Port>
      </NetworkProtocols>
    </GetNetworkProtocolsResponse>
  </env:Body>
</env:Envelope>
```

B.3.4.2 SetNetworkProtocols

REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:tt="http://www.onvif.org/ver10/schema">
  <soap12:Body>
    <SetNetworkProtocols xmlns="http://www.onvif.org/ver10/device/wsdl">
      <NetworkProtocols>
        <tt:Name>HTTP</tt:Name>
        <tt:Enabled>true</tt:Enabled>
        <tt:Port>80</tt:Port>
      </NetworkProtocols>
      <NetworkProtocols>
        <tt:Name>RTSP</tt:Name>
        <tt:Enabled>true</tt:Enabled>
        <tt:Port>554</tt:Port>
      </NetworkProtocols>
```

```
      <NetworkProtocols>
        <tt:Name>HTTPS</tt:Name>
        <tt:Enabled>true</tt:Enabled>
        <tt:Port>443</tt:Port>
      </NetworkProtocols>
    </SetNetworkProtocols>  </soap12:Body>
</soap12:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <SetNetworkProtocolsResponse xmlns="http://www.onvif.org/ver10/device/wsdl">
    </SetNetworkProtocolsResponse>
  </env:Body>
</env:Envelope>
```

B.4 SOAP Communication Traces for Streaming

B.4.1 SOAP Communication Traces for Using an Existing Profile for Media Streaming

The following traces refer to Section 7.1.

B.4.1.1 GetProfiles

SOAP REQUEST

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">

   <soap:Body>
      <trt:GetProfiles/>
   </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
  <SOAP-ENV:Body>
    <<trt:GetProfilesResponse>
      <trt:Profiles fixed="true" token="Profile1">
        <tt:Name>Profile1</tt:Name>
        <tt:VideoSourceConfiguration xsi:type="tt:VideoSourceConfiguration"
token="video_source_config">
          <tt:Name>video_source_config</tt:Name>
          <tt:UseCount>1</tt:UseCount>
          <tt:SourceToken>video_source</tt:SourceToken>
          <tt:Bounds height="720" width="1280" y="1" x="1">
          </tt:Bounds>
        </tt:VideoSourceConfiguration>
        <tt:VideoEncoderConfiguration xsi:type="tt:VideoEncoderConfiguration"
token="video_encoder_config1">
          <tt:Name>video_encoder_config1</tt:Name>
          <tt:UseCount>1</tt:UseCount>
          <tt:Encoding>H264</tt:Encoding>
          <tt:Resolution>
            <tt:Width>1280</tt:Width>
            <tt:Height>720</tt:Height>
          </tt:Resolution>
          <tt:Quality>7</tt:Quality>
          <tt:RateControl>
            <tt:FrameRateLimit>30</tt:FrameRateLimit>
            <tt:EncodingInterval>0</tt:EncodingInterval>
            <tt:BitrateLimit>2048</tt:BitrateLimit>
          </tt:RateControl>
          <tt:H264>
            <tt:GovLength>30</tt:GovLength>
            <tt:H264Profile>Baseline</tt:H264Profile>
          </tt:H264>
          <tt:Multicast>
            <tt:Address>
              <tt:Type>IPv4</tt:Type>
              <tt:IPv4Address>0.0.0.0</tt:IPv4Address>
            </tt:Address>
```

```
            <tt:Port>0</tt:Port>
            <tt:TTL>3</tt:TTL>
            <tt:AutoStart>false</tt:AutoStart>
          </tt:Multicast>
          <tt:SessionTimeout>PT0S</tt:SessionTimeout>
        </tt:VideoEncoderConfiguration>
      </trt:Profiles>
      <!-- (… other profiles) -->
    </trt:GetProfilesResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4.1.2 GetStreamURI

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl"
 xmlns:tt="http://www.onvif.org/ver10/schema">
  <soap:Body>
    <trt:GetStreamUri>
      <trt:StreamSetup>
        <tt:Stream>RTP-Unicast</tt:Stream>
        <tt:Transport>
          <tt:Protocol>UDP</tt:Protocol>
        </Transport>
      </trt:StreamSetup>
      <trt:ProfileToken>Profile1</trt:ProfileToken>
    </trt:GetStreamUri>
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
  <SOAP-ENV:Body>
    <trt:GetStreamUriResponse>
      <trt:MediaUri>
        <tt:Uri>rtsp://192.168.0.100/media/video1</tt:Uri>
        <tt:InvalidAfterConnect>false</tt:InvalidAfterConnect>
        <tt:InvalidAfterReboot>false</tt:InvalidAfterReboot>
        <tt:Timeout>PT100S</tt:Timeout>
      </trt:MediaUri>
    </trt:GetStreamUriResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4.2 SOAP Communication Traces for Media Profile Configuration

The following traces refer to Section 7.2.

B.4.2.1 GetVideoEncoderConfigurations

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
  <soap:Body>
    <trt:GetVideoEncoderConfigurations />
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
  <SOAP-ENV:Body>
    <trt:GetVideoEncoderConfigurationsResponse>
      <trt:Configurations xsi:type="tt:VideoEncoderConfiguration"
token="video_encoder_config1">
        <tt:Name>video_encoder_config1</tt:Name>
        <tt:UseCount>1</tt:UseCount>
        <tt:Encoding>H264</tt:Encoding>
        <tt:Resolution>
          <tt:Width>1280</tt:Width>
          <tt:Height>720</tt:Height>
        </tt:Resolution>
        <tt:Quality>7</tt:Quality>
        <tt:RateControl>
          <tt:FrameRateLimit>30</tt:FrameRateLimit>
          <tt:EncodingInterval>0</tt:EncodingInterval>
          <tt:BitrateLimit>2048</tt:BitrateLimit>
        </tt:RateControl>
        <tt:H264>
          <tt:GovLength>30</tt:GovLength>
          <tt:H264Profile>Baseline</tt:H264Profile>
        </tt:H264>
        <tt:Multicast>
          <tt:Address>
            <tt:Type>IPv4</tt:Type>
            <tt:IPv4Address>0.0.0.0</tt:IPv4Address>
          </tt:Address>
          <tt:Port>0</tt:Port>
          <tt:TTL>3</tt:TTL>
          <tt:AutoStart>false</tt:AutoStart>
        </tt:Multicast>
        <tt:SessionTimeout>PT0S</tt:SessionTimeout>
      </trt:Configurations>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4.2.2 GetVideoEncoderConfigurationOptions

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
  <soap:Body>
    <trt:GetVideoEncoderConfigurationOptions>
      <trt:ProfileToken>Profile1</trt:ProfileToken>
    </trt:GetVideoEncoderConfigurationOptions>
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
  <SOAP-ENV:Body>
    <trt:GetVideoEncoderConfigurationOptionsResponse>
      <trt:Options>
        <tt:QualityRange>
          <tt:Min>1</tt:Min>
          <tt:Max>10</tt:Max>
        </tt:QualityRange>
        <tt:JPEG>
          <tt:ResolutionsAvailable>
            <tt:Width>320</tt:Width>
            <tt:Height>192</tt:Height>
          </tt:ResolutionsAvailable>
          ...(other resolution)
          <tt:ResolutionsAvailable>
            <tt:Width>1280</tt:Width>
            <tt:Height>720</tt:Height>
          </tt:ResolutionsAvailable>
          <tt:FrameRateRange>
            <tt:Min>1</tt:Min>
            <tt:Max>30</tt:Max>
          </tt:FrameRateRange>
          <tt:EncodingIntervalRange>
            <tt:Min>1</tt:Min>
            <tt:Max>1</tt:Max>
          </tt:EncodingIntervalRange>
        </tt:JPEG>
        ...(other codec)
        <tt:Extension>
          <tt:JPEG>
            <tt:BitrateRange>
              <tt:Min>384</tt:Min>
              <tt:Max>2048</tt:Max>
            </tt:BitrateRange>
          </tt:JPEG>
        </tt:Extension>
      </trt:Options>
    </trt:GetVideoEncoderConfigurationOptionsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4.2.3 SetVideoEncoderConfiguration

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl"
 xmlns:tt="http://www.onvif.org/ver10/schema">
  <soap:Body>
    <trt:SetVideoEncoderConfiguration>
      <trt:Configuration token="video_encoder_config1">
        <tt:Name>video_encoder_config1</tt:Name>
        <tt:UseCount>1</tt:UseCount>
        <tt:Encoding>JPEG</tt:Encoding>
        <tt:Resolution>
          <tt:Width>320</tt:Width>
          <tt:Height>192</tt:Height>
        </tt:Resolution>
        <tt:Quality>1</tt:Quality>
        <tt:RateControl>
          <tt:FrameRateLimit>1</tt:FrameRateLimit>
          <tt:EncodingInterval>1</tt:EncodingInterval>
          <tt:BitrateLimit>384</tt:BitrateLimit>
        </tt:RateControl>
        <tt:Multicast>
          <tt:Address>
            <tt:Type>IPv4</tt:Type>
            <tt:IPv4Address>0.0.0.0</tt:IPv4Address>
          </tt:Address>
          <tt:Port>0</tt:Port>
          <tt:TTL>3</tt:TTL>
          <tt:AutoStart>false</tt:AutoStart>
        </tt:Multicast>
        <tt:SessionTimeout>PT0S</tt:SessionTimeout>
      </trt:Configurations>
      <trt:ForcePersistence>true</trt:ForcePersistence>
    </trt:SetVideoEncoderConfiguration>
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
  <SOAP-ENV:Body>
    <trt:SetVideoEncoderConfigurationResponse>
    </trt:SetVideoEncoderConfigurationResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4.3 SOAP Communication Traces for Creating a New Media Profile and Adding an Entity

The following traces refer to Section 7.3.

B.4.3.1 CreateProfile

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
  <soap:Body>
    <trt:CreateProfile>
      <trt:Name>Test Profile</trt:Name>
      <trt:Token>testprof0</trt:Token>
    </trt:CreateProfile>
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
  <SOAP-ENV:Body>
    <trt:CreateProfileResponse>
      <trt:Profile fixed="false" token="testprof0">
        <tt:Name>Test Profile</tt:Name>
      </trt:Profile>
    </trt:CreateProfileResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4.3.2 GetVideoSourceConfigurations

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
  <soap:Body>
    <trt:GetVideoSourceConfigurations />
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
  <SOAP-ENV:Body>
    <trt:GetVideoSourceConfigurationsResponse>
      <trt:Configurations xsi:type="tt:VideoSourceConfiguration"
token="video_source_config">
        <tt:Name>video_source_config</tt:Name>
        <tt:UseCount>2</tt:UseCount>
        <tt:SourceToken>video_source</tt:SourceToken>
        <tt:Bounds height="720" width="1280" y="1" x="1">
        </tt:Bounds>
```

```
      </trt:Configurations>
    </trt:GetVideoSourceConfigurationsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4.3.3 AddVideoSourceConfiguration

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl"
 xmlns:tt="http://www.onvif.org/ver10/schema">
  <soap:Body>
    <trt:AddVideoSourceConfiguration>
      <trt:ProfileToken>testprof0</trt:ProfileToken>
      <trt:ConfigurationToken>video_source_config</trt:ConfigurationToken>
    </trt:AddVideoSourceConfiguration>
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
  <SOAP-ENV:Body>
    <trt:AddVideoSourceConfigurationResponse>
    </trt:AddVideoSourceConfigurationResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4.3.4 AddVideoEncoderConfiguration

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
  <soap:Body>
    <trt:AddVideoEncoderConfiguration>
      <trt:ProfileToken>testprof0</trt:ProfileToken>
      <trt:ConfigurationToken>video_encoder_config1</trt:ConfigurationToken>
    </trt:AddVideoEncoderConfiguration>
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
  <SOAP-ENV:Body>
    <trt:AddVideoEncoderConfigurationResponse>
    </trt:AddVideoEncoderConfigurationResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4.4 SOAP Communication Traces for Multicast Streaming

The following traces refer to Section 7.4.

B.4.4.1 StartMulticastStreaming

REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:tt="http://www.onvif.org/ver10/schema">
  <soap12:Body>
    <StartMulticastStreaming xmlns="http://www.onvif.org/ver10/media/wsdl">
      <ProfileToken>profile_1</ProfileToken>
    </StartMulticastStreaming>
  </soap12:Body>
</soap12:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <StartMulticastStreamingResponse xmlns="http://www.onvif.org/ver10/media/wsdl">
    </StartMulticastStreamingResponse>
  </env:Body>
</env:Envelope>
```

B.4.4.2 StopMulticastStreaming

REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:tt="http://www.onvif.org/ver10/schema">
  <soap12:Body>
    <StopMulticastStreaming xmlns="http://www.onvif.org/ver10/media/wsdl">
      <ProfileToken>profile_1</ProfileToken>
    </StopMulticastStreaming>
  </soap12:Body>
</soap12:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <StopMulticastStreamingResponse xmlns="http://www.onvif.org/ver10/media/wsdl">
    </StopMulticastStreamingResponse>
  </env:Body>
</env:Envelope>
```

B.4.5 RTSP Communication Traces for Audio Backchannel Handling

The following traces refer to Section 7.5.

B.4.5.1 RTSP Communication Trace for RTSP Session Setup Example

In this example, the clients sends a `DESCRIBE` message. It includes the ONVIF-defined "Require tag" to indicate that it wants to include an audio backchannel connection. The response of the device contains the `sdp` file that describes the streams that are sent from the device, as well as all stream types that can be decoded from the device.

Request	Response
DESCRIBE rtsp://192.168.0.1 RTSP/1.0 CSeq: 1 User-Agent: ONVIF Rtsp client Accept: application/sdp Require: www.onvif.org/ver20/backchannel	RTSP/1.0 200 OK CSeq: 1 Content-Type: application/sdp Content-Length: xxx v=0 o= 2890842807 IN IP4 192.168.0.1 s=RTSP Session with audiobackchannel m=video 0 RTP/AVP 26 a=control:rtsp://192.168.0.1/video a=recvonly m=audio 0 RTP/AVP 0 a=control:rtsp://192.168.0.1/audio a=recvonly m=audio 0 RTP/AVP 0 a=control:rtsp://192.168.0.1/G711_audiobackchannel a=rtpmap:0 PCMU/8000 a=sendonly m=audio 98 RTP/AVP 0 a=control:rtsp://192.168.0.1/G726_audiobackchannel a=rtpmap:98 G726-16/8000 a=sendonly m=audio 0 RTP/AVP 97 a=control:rtsp://192.168.0.1/AAC_audiobackchannel a=rtpmap:97 MPEG4-GENERIC/11025/1 a=fmtp:97 profile-level-id=1;mode=AAC-hbr; sizelength=13;indexlength=3;indexdeltalength=3;config=1508 a=sendonly

In the next step, the client sets up the video and audio downstreams.

Request	Response
SETUP rtsp://192.168.0.1/video RTSP/1.0 CSeq: 2 Transport: RTP/AVP;unicast;interleaved=0-1	RTSP/1.0 200 OK CSeq: 2 Session: 123124;timeout=60 Transport:RTP/AVP;unicast;interleaved=0-1

Request	Response
SETUP rtsp://192.168.0.1/audio RTSP/1.0 CSeq: 3 Session: 123124 Transport: RTP/AVP;unicast;interleaved=2-3	RTSP/1.0 200 OK CSeq: 3 Session: 123124;timeout=60 Transport:RTP/AVP;unicast;interleaved=2-3

Then the client establishes the G.711 audio upstream by sending a `SETUP` request. It uses the RTSP control URL for the G.711 stream from the `sdp` file.

Request	Response
SETUP rtsp://192.168.0.1/G711_audioback RTSP/1.0 CSeq: 4 Session: 123124 Transport: RTP/AVP;unicast;interleaved=4-5 Require: www.onvif.org/ver20/backchannel	RTSP/1.0 200 OK CSeq: 4 Session: 123124;timeout=60 Transport:RTP/AVP;unicast;interleaved=4-5

The client now can start the complete RTSP session by sending a `PLAY` request.

Request	Response
PLAY rtsp://192.168.0.1 RTSP/1.0 CSeq: 5 Session: 123124 Require: www.onvif.org/ver20/backchannel	RTSP/1.0 200 OK CSeq: 5 Session: 123124;timeout=60

After receiving the PLAY response, the client can send audio data to the device. It uses the RTSP channel 4 as indicated during session setup. For RTSP over HTTP, the client uses the same socket as for the RTSP requests.

B.4.6 SOAP Communication Traces for Setting Up Metadata Streaming

The following traces refer to Section 7.6.

B.4.6.1 media:GetProfiles

Request: media.**GetProfiles**

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:media="http://www.onvif.org/ver10/media/wsdl">
    <SOAP-ENV:Body>
    <media:GetProfiles/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response to media.GetProfiles

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
<SOAP-ENV:Body>
  <trt:GetProfilesResponse>
    <trt:Profiles token="quality_h264">
      <tt:Name>quality h264</tt:Name>
        ..
      <tt:VideoEncoderConfiguration token="quality_h264">
        ..
      </tt:VideoEncoderConfiguration>
    </trt:Profiles>
    .. (some profiles removed)
    <trt:Profiles token="metadata">
      <tt:Name>metadata apg</tt:Name>
      <tt:MetadataConfiguration token="0">
        <tt:Name>metadata</tt:Name>
        <tt:UseCount>1</tt:UseCount>
        <tt:Events>
          <tt:Filter>
            <wsnt:TopicExpression
Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">t
ns1:Device//.</wsnt:TopicExpression>
          </tt:Filter>
        </tt:Events>
        <tt:Multicast>
          <tt:Address>
            <tt:Type>IPv4</tt:Type><tt:IPv4Address>0.0.0.0</tt:IPv4Address>
          </tt:Address>
          <tt:Port>0</tt:Port>
          <tt:TTL>1</tt:TTL>
          <tt:AutoStart>false</tt:AutoStart>
        </tt:Multicast>
        <tt:SessionTimeout>PT60S</tt:SessionTimeout>
      </tt:MetadataConfiguration>
    </trt:Profiles>
  </trt:GetProfilesResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4.6.2 media:GetMetadataConfigurations

Request: media.**GetMetadataConfigurations**

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:SOAP-ENC="http://www.w3.org/2003/05/soap-encoding"
 xmlns:media="http://www.onvif.org/ver10/media/wsdl">

<SOAP-ENV:Body>
    <media:GetMetadataConfigurations/>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response to media.GetMetadataConfigurations

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:SOAP-ENC="http://www.w3.org/2003/05/soap-encoding"
xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
xmlns:trt="http://www.onvif.org/ver10/media/wsdl">

<SOAP-ENV:Body>
  <trt:GetMetadataConfigurationsResponse>
    <trt:Configurations token="0">
      <tt:Name>metadata</tt:Name>
      <tt:UseCount>1</tt:UseCount>
      <tt:Events>
        <tt:Filter>
          <wsnt:TopicExpression
Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
>tns1:Device//.</wsnt:TopicExpression>
        </tt:Filter>
      </tt:Events>
      <tt:Multicast>
        <tt:Address>
          <tt:Type>IPv4</tt:Type>
          <tt:IPv4Address>0.0.0.0</tt:IPv4Address>
        </tt:Address>
        <tt:Port>0</tt:Port>
        <tt:TTL>1</tt:TTL>
        <tt:AutoStart>false</tt:AutoStart>
      </tt:Multicast>
      <tt:SessionTimeout>PT60S</tt:SessionTimeout>
    </trt:Configurations>
    <trt:Configurations token="1">
      .. (additional configuration removed)
    </trt:Configurations>
     .. (additional configuration removed)
  </trt:GetMetadataConfigurationsResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4.6.3 media:GetMetadataConfigurationOptions

Request: media.**GetMetadataConfigurationOptions**

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:media="http://www.onvif.org/ver10/media/wsdl">
<SOAP-ENV:Body>
    <media:GetMetadataConfigurationOptions/>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response to media.GetMetadataConfigurationOptions

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
<SOAP-ENV:Body>
  <trt:GetMetadataConfigurationOptionsResponse>
    <trt:Options>
      <tt:PTZStatusFilterOptions>
        <tt:PanTiltStatusSupported>false</tt:PanTiltStatusSupported>
        <tt:ZoomStatusSupported>false</tt:ZoomStatusSupported>
      </tt:PTZStatusFilterOptions>
    </trt:Options>
  </trt:GetMetadataConfigurationOptionsResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4.6.4 media:GetMetadataConfiguration

Request: media.**GetMetadataConfiguration**

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:media="http://www.onvif.org/ver10/media/wsdl">
<SOAP-ENV:Body>
   <media:GetMetadataConfiguration>
     <media:ConfigurationToken>0</media:ConfigurationToken>
   </media:GetMetadataConfiguration>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response to media.GetMetadataConfiguration

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
xmlns:trt="http://www.onvif.org/ver10/media/wsdl"
xmlns:tns1="http://www.onvif.org/ver10/topics"
xmlns:tnsvendor="http://www.vendor.com/2009/event/topics">
<SOAP-ENV:Body>
  <trt:GetMetadataConfigurationResponse>
    <trt:Configuration token="0">
      <tt:Name>metadata</tt:Name>
      <tt:UseCount>1</tt:UseCount>
      <tt:Events>
        <tt:Filter>
          <wsnt:TopicExpression
Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
>tns1:Device//.</wsnt:TopicExpression>
        </tt:Filter>
      </tt:Events>
      <tt:Multicast>
        <tt:Address>
          <tt:Type>IPv4</tt:Type>
          <tt:IPv4Address>0.0.0.0</tt:IPv4Address>
        </tt:Address>
        <tt:Port>0</tt:Port>
        <tt:TTL>1</tt:TTL>
        <tt:AutoStart>false</tt:AutoStart>
      </tt:Multicast>
      <tt:SessionTimeout>PT60S</tt:SessionTimeout>
    </trt:Configuration>
```

```
  </trt:GetMetadataConfigurationResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4.6.5 media:SetMetadataConfiguration

Request: media.**SetMetadataConfiguration**

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
 xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
 xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:media="http://www.onvif.org/ver10/media/wsdl"
 xmlns:tns1="http://www.onvif.org/ver10/topics"
 xmlns:tnsvendor="http://www.vendor.com/2009/event/topics">
    <SOAP-ENV:Body>
      <media:SetMetadataConfiguration>
        <media:Configuration token="0">
          <tt:Name>metadata</tt:Name>
          <tt:UseCount>1</tt:UseCount>
          <tt:Events>
            <tt:Filter>
              <wsnt:TopicExpression
Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet"
>tns1:Device//.</wsnt:TopicExpression>
            </tt:Filter>
          </tt:Events>
          <tt:Multicast>
            <tt:Address>
              <tt:Type>IPv4</tt:Type>
              <tt:IPv4Address>0.0.0.0</tt:IPv4Address>
            </tt:Address>
            <tt:Port>0</tt:Port>
            <tt:TTL>1</tt:TTL>
          <tt:AutoStart>false</tt:AutoStart>
        </tt:Multicast>
        <tt:SessionTimeout>PT60S</tt:SessionTimeout>
      </media:Configuration>
      <media:ForcePersistence>false</media:ForcePersistence>
    </media:SetMetadataConfiguration>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response to media.SetMetadataConfiguration

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">

<SOAP-ENV:Body>
    <trt:SetMetadataConfigurationResponse/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4.6.6 media:GetProfile

Request: media.**GetProfile**

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:SOAP-ENC="http://www.w3.org/2003/05/soap-encoding"
 xmlns:media="http://www.onvif.org/ver10/media/wsdl">
<SOAP-ENV:Body>
    <media:GetProfile>
      <media:ProfileToken>metadata</media:ProfileToken>
    </media:GetProfile>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response to media.GetProfile

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
xmlns:trt="http://www.onvif.org/ver10/media/wsdl"
xmlns:tns1="http://www.onvif.org/ver10/topics">
<SOAP-ENV:Body>
  <trt:GetProfileResponse>
    <trt:Profile token="metadata">
      <tt:Name>metadata apg</tt:Name>
      <tt:MetadataConfiguration token="0">
        <tt:Name>metadata</tt:Name>
        <tt:UseCount>1</tt:UseCount>
        <tt:Events>
          <tt:Filter>
            <wsnt:TopicExpression
Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSe
t">tns1:Device//.</wsnt:TopicExpression>
          </tt:Filter>
        </tt:Events>
        <tt:Multicast>
          <tt:Address>
            <tt:Type>IPv4</tt:Type>
            <tt:IPv4Address>0.0.0.0</tt:IPv4Address>
          </tt:Address>
          <tt:Port>0</tt:Port>
          <tt:TTL>1</tt:TTL>
          <tt:AutoStart>false</tt:AutoStart>
        </tt:Multicast>
        <tt:SessionTimeout>PT60S</tt:SessionTimeout>
      </tt:MetadataConfiguration>
    </trt:Profile>
  </trt:GetProfileResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.4.6.7 media:GetStreamUri

Request: media.**GetStreamUri**

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tt="http://www.onvif.org/ver10/schema"
```

```
 xmlns:media="http://www.onvif.org/ver10/media/wsdl">
  <SOAP-ENV:Body>
    <media:GetStreamUri><media:StreamSetup>
      <tt:Stream>RTP-Unicast</tt:Stream>
      <tt:Transport><tt:Protocol>RTSP</tt:Protocol>
      </tt:Transport></media:StreamSetup>
    <media:ProfileToken>metadata</media:ProfileToken>
   </media:GetStreamUri>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response to media.GetSTreamUri

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
<SOAP-ENV:Body>
  <trt:GetStreamUriResponse>
    <trt:MediaUri>
      <tt:Uri>rtsp://169.254.76.145/onvif-
media/media.amp?profile=metadata&amp;sessiontimeout=60</tt
:Uri>
      <tt:InvalidAfterConnect>true</tt:InvalidAfterConnect>
      <tt:InvalidAfterReboot>false</tt:InvalidAfterReboot>
      <tt:Timeout>PT0S</tt:Timeout>
    </trt:MediaUri>
  </trt:GetStreamUriResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.5 SOAP Communication Traces for Controlling

B.5.1 SOAP Communication Trace for Adding a PTZ Configuration into a Media Profile

The following trace refers to Section 8.1.

B.5.1.1 GetConfigurations

SOAP REQUEST

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl">
  <SOAP-ENV:Body>
    <tptz:GetConfigurationsResponse>
      <tptz:PTZConfiguration xsi:type="tt:PTZConfiguration" token="1">
        <tt:Name>default</tt:Name>
        <tt:UseCount>0</tt:UseCount>
        <tt:NodeToken>1</tt:NodeToken>
        <tt:DefaultAbsolutePantTiltPositionSpace>
http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace
</tt:DefaultAbsolutePantTiltPositionSpace>
        <tt:DefaultAbsoluteZoomPositionSpace>
http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace
</tt:DefaultAbsoluteZoomPositionSpace>
        <tt:DefaultRelativePanTiltTranslationSpace>
http://www.onvif.org/ver10/tptz/PanTiltSpaces/TranslationGenericSpace
</tt:DefaultRelativePanTiltTranslationSpace>
        <tt:DefaultRelativeZoomTranslationSpace>
http://www.onvif.org/ver10/tptz/ZoomSpaces/TranslationGenericSpace
</tt:DefaultRelativeZoomTranslationSpace>
        <tt:DefaultContinuousPanTiltVelocitySpace>
http://www.onvif.org/ver10/tptz/PanTiltSpaces/VelocityGenericSpace
</tt:DefaultContinuousPanTiltVelocitySpace>
        <tt:DefaultContinuousZoomVelocitySpace>
http://www.onvif.org/ver10/tptz/ZoomSpaces/VelocityGenericSpace
</tt:DefaultContinuousZoomVelocitySpace>
        <tt:DefaultPTZSpeed>
          <tt:PanTilt
space="http://www.onvif.org/ver10/tptz/PanTiltSpaces/GenericSpeedSpace"
 y="1" x="1">
          </tt:PanTilt>
          <tt:Zoom
 space="http://www.onvif.org/ver10/tptz/ZoomSpaces/ZoomGenericSpeedSpace"
 x="1">
          </tt:Zoom>
        </tt:DefaultPTZSpeed>
        <tt:DefaultPTZTimeout>PT60S</tt:DefaultPTZTimeout>
      </tptz:PTZConfiguration>
    </tptz:GetConfigurationsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.5.1.2 AddPTZConfiguration

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:trt="http://www.onvif.org/ver20/media/wsdl">
  <soap:Body>
    <trt:AddPTZConfiguration>
      <trt:ProfileToken>Profile1</trt:ProfileToken>
      <trt:ConfigurationToken>1</trt:ConfigurationToken>
    </trt:AddPTZConfiguration>
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
  <SOAP-ENV:Body>
    <trt:AddPTZConfigurationResponse>
    </trt:AddPTZConfigurationResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.5.2 SOAP Communication Traces for Changing a PTZ Configuration

The following traces refer to Section 8.2.

B.5.2.1 GetConfigurationOptions

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl">
  <soap:Body>
    <tptz:GetConfigurationOptions>
      <tptz:ConfigurationToken>1</tptz:ConfigurationToken>
    </tptz:GetConfigurationOptions>
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tt="http://www.onvif.org/ver10/schema"
 xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl">
  <SOAP-ENV:Body>
    <tptz:GetConfigurationOptionsResponse>
      <tptz:PTZConfigurationOptions>
        <tt:Spaces>
          <tt:AbsolutePanTiltPositionSpace>
            <tt:URI>
http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace</tt:URI>
            <tt:XRange>
              <tt:Min>-1</tt:Min>
              <tt:Max>1</tt:Max>
            </tt:XRange>
            <tt:YRange>
```

```
              <tt:Min>-1</tt:Min>
              <tt:Max>1</tt:Max>
            </tt:YRange>
          </tt:AbsolutePanTiltPositionSpace>
          <tt:AbsolutePanTiltPositionSpace>
            <tt:URI>
http://www.onvif.org/ver10/tptz/PanTiltSpaces/SphericalPositionSpace</tt:URI>
            <tt:XRange>
              <tt:Min>-90</tt:Min>
              <tt:Max>0</tt:Max>
            </tt:XRange>
            <tt:YRange>
              <tt:Min>-90</tt:Min>
              <tt:Max>0</tt:Max>
            </tt:YRange>
          </tt:AbsolutePanTiltPositionSpace>
          <tt:AbsoluteZoomPositionSpace>
            <tt:URI>
http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace</tt:URI>
            <tt:XRange>
              <tt:Min>0</tt:Min>
              <tt:Max>1</tt:Max>
            </tt:XRange>
          </tt:AbsoluteZoomPositionSpace>
          (... other space )


          <tt:PanTiltSpeedSpace>
            <tt:URI>
http://www.onvif.org/ver10/tptz/PanTiltSpaces/GenericSpeedSpace</tt:URI>
            <tt:XRange>
              <tt:Min>0</tt:Min>
              <tt:Max>1</tt:Max>
            </tt:XRange>
          </tt:PanTiltSpeedSpace>
          <tt:PanTiltSpeedSpace>
            <tt:URI>
http://www.onvif.org/ver10/tptz/PanTiltSpaces/SpeedSpaceDegrees</tt:URI>
            <tt:XRange>
              <tt:Min>0</tt:Min>
              <tt:Max>90</tt:Max>
            </tt:XRange>
          </tt:PanTiltSpeedSpace>
          <tt:ZoomSpeedSpace>
            <tt:URI>
http://www.onvif.org/ver10/tptz/ZoomSpaces/ZoomGenericSpeedSpace</tt:URI>
            <tt:XRange>
              <tt:Min>0</tt:Min>
              <tt:Max>1</tt:Max>
            </tt:XRange>
          </tt:ZoomSpeedSpace>
        </tt:Spaces>
        <tt:PTZTimeout>
          <tt:Min>PT0S</tt:Min>
          <tt:Max>PT2H</tt:Max>
        </tt:PTZTimeout>
      </tptz:PTZConfigurationOptions>
    </tptz:GetConfigurationOptionsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.5.2.2 SetConfiguration

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl"
 xmlns:tt="http://www.onvif.org/ver10/schema">
  <soap:Body>
    <tptz:SetConfiguration>
      <tptz:PTZConfiguration token="1">
        <tt:Name>default</tt:Name>
        <tt:UseCount>0</tt:UseCount>
        <tt:NodeToken>1</tt:NodeToken>
        <tt:DefaultAbsolutePantTiltPositionSpace>
http://www.onvif.org/ver10/tptz/PanTiltSpaces/SphericalPositionSpace
</tt:DefaultAbsolutePantTiltPositionSpace>
        <tt:DefaultAbsoluteZoomPositionSpace>
http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace
</tt:DefaultAbsoluteZoomPositionSpace>
        <tt:DefaultRelativePanTiltTranslationSpace>
http://www.onvif.org/ver10/tptz/PanTiltSpaces/TranslationGenericSpace
</tt:DefaultRelativePanTiltTranslationSpace>
        <tt:DefaultRelativeZoomTranslationSpace>
http://www.onvif.org/ver10/tptz/ZoomSpaces/TranslationGenericSpace
</tt:DefaultRelativeZoomTranslationSpace>
        <tt:DefaultContinuousPanTiltVelocitySpace>
http://www.onvif.org/ver10/tptz/PanTiltSpaces/VelocityGenericSpace
</tt:DefaultContinuousPanTiltVelocitySpace>
        <tt:DefaultContinuousZoomVelocitySpace>
http://www.onvif.org/ver10/tptz/ZoomSpaces/VelocityGenericSpace
</tt:DefaultContinuousZoomVelocitySpace>
        <tt:DefaultPTZSpeed>
          <tt:PanTilt
space="http://www.onvif.org/ver10/tptz/PanTiltSpaces/SpeedSpaceDegrees"
 y="90" x="90">
          </tt:PanTilt>
          <tt:Zoom
 space="http://www.onvif.org/ver10/tptz/ZoomSpaces/ZoomGenericSpeedSpace"
 x="1">
          </tt:Zoom>
        </tt:DefaultPTZSpeed>
        <tt:DefaultPTZTimeout>PT60S</tt:DefaultPTZTimeout>
      </tptz:PTZConfiguration>
      <tptz:ForcePersistence>true</tptz:ForcePersistence>
    </tptz:SetConfiguration>
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl">
  <SOAP-ENV:Body>
    <tptz:SetConfigurationResponse>
    </tptz:SetConfigurationResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.5.3 SOAP Communication Traces for Move Operation

The following traces refer to Section 8.3.

B.5.3.1 ContinuousMove

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl"
 xmlns:tt="http://www.onvif.org/ver10/schema">
  <soap:Body>
    <tptz:ContinuousMove>
      <tptz:ProfileToken>Profile1</tptz:ProfileToken>
      <tptz:Velocity>
        <tt:PanTilt x="1" y="1"/>
        <tt:Zoom x="1"/>
      </tptz:Velocity>
    </tptz:ContinuousMove>
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl">
  <SOAP-ENV:Body>
    <tptz:ContinuousMoveResponse>
    </tptz:ContinuousMoveResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.5.3.2 Stop

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl">
  <soap:Body>
    <tptz:Stop>
      <tptz:ProfileToken>Profile1</tptz:ProfileToken>
      <tptz:PanTilt>true</tptz:PanTilt>
      <tptz:Zoom>true</tptz:Zoom>
    </tptz:Stop>
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl">
  <SOAP-ENV:Body>
    <tptz:StopResponse>
    </tptz:StopResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.5.4 SOAP Communication Traces for Set / Goto Preset Position

The following traces refer to Section 8.4.

B.5.4.1 SetPreset

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl">
  <soap:Body>
    <tptz:SetPreset>
      <tptz:ProfileToken>Profile1</tptz:ProfileToken>
      <tptz:PresetName>PresetName1</tptz:PresetName>
    </tptz:SetPreset>
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl">
  <SOAP-ENV:Body>
    <tptz:SetPresetResponse>
      <tptz:PresetToken>Preset1</tptz:PresetToken>
    </tptz:SetPresetResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.5.4.2 GetPreset

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl">
  <soap:Body>
    <tptz:GetPresets>
      <tptz:ProfileToken>Profile1</tptz:ProfileToken>
    </tptz:GetPresets>
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl">
  <SOAP-ENV:Body>
    <tptz:GetPresetsResponse>
      <tptz:Preset token="Preset1">
        <tt:Name>PresetName1</tt:Name>
        <tt:PTZPosition>
          <tt:PanTilt
space="http://www.onvif.org/ver10/tptz/PanTiltSpaces/PositionGenericSpace"
y="1.33333337" x="-0.884068608">
          </tt:PanTilt>
          <tt:Zoom
space="http://www.onvif.org/ver10/tptz/ZoomSpaces/PositionGenericSpace" x="1.1425662">
```

```
          </tt:Zoom>
        </tt:PTZPosition>
      </tptz:Preset>
    </tptz:GetPresetsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.5.4.3 GotoPreset

SOAP REQUEST

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl">
  <soap:Body>
    <tptz:GotoPreset>
      <tptz:ProfileToken>Profile1</tptz:ProfileToken>
      <tptz:PresetToken>Preset1</tptz:PresetToken>
    </tptz:GotoPreset>
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:tptz="http://www.onvif.org/ver20/ptz/wsdl">
  <SOAP-ENV:Body>
    <tptz:GotoPresetResponse>
    </tptz:GotoPresetResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.6 SOAP Communication Traces for Eventing

B.6.1 SOAP Communication Trace for GetEventProperties

The following trace refers to Section 9.1.1.

B.6.1.1 GetEventProperties

Request events.**GetEventProperties**

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:wsa="http://www.w3.org/2005/08/addressing"
 xmlns:tev="http://www.onvif.org/ver10/events/wsdl">
  <SOAP-ENV:Header><wsa:Action>
http://www.onvif.org/ver10/events/wsdl/EventPortType/GetEventPropertiesRequest
   </wsa:Action>
   <wsa:To>http://169.254.76.145/onvif/services</wsa:To>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tev:GetEventProperties/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response to events.GetEventProperties

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsa5="http://www.w3.org/2005/08/addressing"
xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
xmlns:tev="http://www.onvif.org/ver10/events/wsdl"
xmlns:tns1="http://www.onvif.org/ver10/topics"
xmlns:tnsvendor="http://www.example.com/2009/event/topics">
  <SOAP-ENV:Header>
    <wsa5:Action SOAP-ENV:mustUnderstand="true">http://www.onvif.org/ver10/events/wsdl/
EventPortType/GetEventPropertiesResponse </wsa5:Action>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tev:GetEventPropertiesResponse>
      <tev:TopicNamespaceLocation>http://www.onvif.org/onvif/ver10/topics/topicns.xml</
tev:TopicNamespaceLocation>
      <wsnt:FixedTopicSet>false</wsnt:FixedTopicSet>
      <wstop:TopicSet>
        <tns1:VideoSource wstop:topic="true">
          <tnsvendor:Tampering wstop:topic="true">
            <tt:MessageDescription>
              <tt:Source>
                <tt:SimpleItemDescription Name="channel" Type="xsd:int">
              </tt:SimpleItemDescription>
            </tt:Source>
            <tt:Data>
              <tt:SimpleItemDescription Name="tampering" Type="xsd:int">
            </tt:SimpleItemDescription>
          </tt:Data>
        </tt:MessageDescription>
      </tnsvendor:Tampering>
    </tns1:VideoSource>
    <tns1:Device wstop:topic="true">
      <tnsvendor:IO wstop:topic="true">
        <VirtualPort wstop:topic="true">
          <tt:MessageDescription>
            <tt:Source>
```

```
              <tt:SimpleItemDescription Name="port" Type="xsd:int">
            </tt:SimpleItemDescription>
          </tt:Source>
          <tt:Data>
            <tt:SimpleItemDescription Name="state" Type="xsd:int">
            </tt:SimpleItemDescription>
          </tt:Data>
        </tt:MessageDescription>
      </VirtualPort>
    </tnsvendor:IO>
    <tnsvendor:Sensor wstop:topic="true">
      <PIR wstop:topic="true">
        <tt:MessageDescription>
          <tt:Source>
            <tt:SimpleItemDescription Name="sensor" Type="xsd:int">
            </tt:SimpleItemDescription>
              </tt:Source>
              <tt:Data>
                <tt:SimpleItemDescription Name="state" Type="xsd:int">
                </tt:SimpleItemDescription>
              </tt:Data>
            </tt:MessageDescription>
          </PIR>
        </tnsvendor:Sensor>
      </tns1:Device>
      <tns1:VideoAnalytics wstop:topic="true">
          <tnsvendor:MotionDetection wstop:topic="true">
            <tt:MessageDescription>
              <tt:Source>
                <tt:SimpleItemDescription Name="window" Type="xsd:int">
                </tt:SimpleItemDescription>
              </tt:Source>
              <tt:Data>
                <tt:SimpleItemDescription Name="motion" Type="xsd:int">
                </tt:SimpleItemDescription>
              </tt:Data>
            </tt:MessageDescription>
          </tnsvendor:MotionDetection>
        </tns1:VideoAnalytics>
        <tns1:AudioSource wstop:topic="true">
          <tnsvendor:TriggerLevel wstop:topic="true">
            <tt:MessageDescription>
              <tt:Source>
                <tt:SimpleItemDescription Name="channel" Type="xsd:int">
                </tt:SimpleItemDescription>
              </tt:Source>
              <tt:Data>
                <tt:SimpleItemDescription Name="triggered" Type="xsd:int">
                </tt:SimpleItemDescription>
              </tt:Data>
            </tt:MessageDescription>
          </tnsvendor:TriggerLevel>
        </tns1:AudioSource>
      </wstop:TopicSet>
      <wsnt:TopicExpressionDialect>http://www.onvif.org/ver10/tev/topicExpression/Concr
eteSet</wsnt:TopicExpressionDialect>
      <wsnt:TopicExpressionDialect>http://docs.oasis-open.org/wsn/t-1/TopicExpression/C
oncrete</wsnt:TopicExpressionDialect>
      <tev:MessageContentFilterDialect>http://www.onvif.org/ver10/tev/messageContentFil
ter/ItemFilter</tev:MessageContentFilterDialect>
      <tev:MessageContentSchemaLocation>http://www.onvif.org/ver10/schema/onvif.xsd</te
v:MessageContentSchemaLocation>
    </tev:GetEventPropertiesResponse>
  </SOAP-ENV:Body>
```

```
</SOAP-ENV:Envelope>
```

B.6.2 SOAP Communication Traces for Setting Up PullPoint Subscription

The following traces refer to Section 9.2.

B.6.2.1 CreatePullPointSubscription

Request events.CreatePullPointSubscription

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:wsa="http://www.w3.org/2005/08/addressing"
 xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
 xmlns:tev="http://www.onvif.org/ver10/events/wsdl"
 xmlns:tns1="http://www.onvif.org/ver10/topics"
 xmlns:tnsvendor="http://www.vendor.com/2009/event/topics">
  <SOAP-ENV:Header>
    <wsa:Action>
http://www.onvif.org/ver10/events/wsdl/EventPortType/CreatePullPointSubscriptionRequest
    </wsa:Action>
    <wsa:To>http://169.254.76.145/onvif/services</wsa:To>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tev:CreatePullPointSubscription>
      <tev:Filter>
        <wsnt:TopicExpression Dialect="http://www.onvif.org/ver10/tev/topicExpression/C
oncreteSet">tns1:Device/tnsvendor:IO//.|tns1:Device/tnsvendor:Sensor/PIR</wsnt:TopicExp
ression>
      </tev:Filter>
      <tev:InitialTerminationTime>PT1M</tev:InitialTerminationTime>
    </tev:CreatePullPointSubscription>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response to events.CreatePullPointSubscription

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa5="http://www.w3.org/2005/08/addressing"
xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
xmlns:tev="http://www.onvif.org/ver10/events/wsdl">
  <SOAP-ENV:Header>
    <wsa5:Action SOAP-ENV:mustUnderstand="true">http://www.onvif.org/ver10/events/wsdl/
EventPortType/CreatePullPointSubscriptionResponse</wsa5:Action>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tev:CreatePullPointSubscriptionResponse>
      <tev:SubscriptionReference>
        <wsa5:Address>http://192.168.1.24/onvif/services</wsa5:Address>
        <wsa5:ReferenceParameters>
          <dom0:SubscriptionId xmlns:dom0="http://www.example.com/2009/event">6</dom0:S
ubscriptionId>
        </wsa5:ReferenceParameters>
      </tev:SubscriptionReference>
      <wsnt:CurrentTime>2010-10-29T15:52:30Z</wsnt:CurrentTime>
      <wsnt:TerminationTime>2010-10-29T15:53:30Z</wsnt:TerminationTime>
    </tev:CreatePullPointSubscriptionResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.6.2.2 PullMessages

Request: events.PullMessages

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:wsa="http://www.w3.org/2005/08/addressing"
 xmlns:tev="http://www.onvif.org/ver10/events/wsdl">
 <SOAP-ENV:Header>
    <wsa:Action>http://www.onvif.org/ver10/events/wsdl/PullPointSubscription/
PullMessagesRequest</wsa:Action>
    <wsa:To>http://192.168.1.24/onvif/services</wsa:To>
    <dom0:SubscriptionId
xmlns:dom0="http://www.example.com/2009/event">6</dom0:SubscriptionId>
 </SOAP-ENV:Header>
 <SOAP-ENV:Body>
   <tev:PullMessages>
     <tev:Timeout>PT5S</tev:Timeout>
     <tev:MessageLimit>1</tev:MessageLimit>
   </tev:PullMessages>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response – on success and messages available in time (request 1 and 2 of example)

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa5="http://www.w3.org/2005/08/addressing"
xmlns:tt="http://www.onvif.org/ver10/schema"
xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2"
xmlns:tev="http://www.onvif.org/ver10/events/wsdl"
xmlns:tns1="http://www.onvif.org/ver10/topics"
xmlns:tnsvendor="http://www.vendor.com/2009/event/topics">
  <SOAP-ENV:Header>
    <wsa5:Action SOAP-ENV:mustUnderstand="true" >http://www.onvif.org/ver10/events/wsd
l/PullPointSubscription/PullMessagesResponse</wsa5:Action>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tev:PullMessagesResponse>
      <tev:CurrentTime>2010-10-29T15:52:36Z</tev:CurrentTime>
      <tev:TerminationTime>2010-10-29T15:53:30Z</tev:TerminationTime>
      <wsnt:NotificationMessage>
        <wsnt:Topic Dialect="http://docs.oasis-open.org/wsn/t-
1/TopicExpression/Simple">tns1:Device/tnsvendor:IO/VirtualPort</wsnt:Topic>
        <wsnt:ProducerReference>
          <wsa5:Address>uri://a1f48ac2-dc8b-11df-b255-
00408c1836b2/ProducerReference</wsa5:Address>
        </wsnt:ProducerReference>
        <wsnt:Message>
          <tt:Message UtcTime="2010-10-29T15:52:30Z" PropertyOperation="Initialized">
            <tt:Source>
              <tt:SimpleItem Name="port" Value="0">
              </tt:SimpleItem>
            </tt:Source>
            <tt:Key/>
            <tt:Data>
              <tt:SimpleItem Name="state" Value="0">
              </tt:SimpleItem>
            </tt:Data>
          </tt:Message>
        </wsnt:Message>
      </wsnt:NotificationMessage>
    </tev:PullMessagesResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response – on success but no messages available (request 3 of example)

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa5="http://www.w3.org/2005/08/addressing"
xmlns:tev="http://www.onvif.org/ver10/events/wsdl">
  <SOAP-ENV:Header>
    <wsa5:Action SOAP-ENV:mustUnderstand="true">http://docs.oasis-open.org/wsn/bw-
2/NotificationProducer/PullMessagesResponse</wsa5:Action>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <tev:PullMessagesResponse>
      <tev:CurrentTime>2010-10-29T15:52:41Z</tev:CurrentTime>
      <tev:TerminationTime>2010-10-29T15:53:30Z</tev:TerminationTime>
    </tev:PullMessagesResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.6.3 SOAP Communication Trace for Setting Up WS-BaseNotification

The following trace refers to Section 9.3.

B.6.3.1 Subscribe

Request events.Subscribe

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
 xmlns:wsa="http://www.w3.org/2005/08/addressing"
 xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2">
  <SOAP-ENV:Header>
    <wsa:Action>http://docs.oasis-open.org/wsn/bw-
2/NotificationProducer/SubscribeRequest</wsa:Action>
    <wsa:To>http://169.254.232.42/onvif/services</wsa:To>
  </SOAP-ENV:Header>
 <SOAP-ENV:Body>
    <wsnt:Subscribe>
  <wsnt:ConsumerReference>
<wsa:Address>http://10.96.6.4:8000/consumerinterface?from=192.168.1.16%26serno=00408C18
37C1</wsa:Address>
  </wsnt:ConsumerReference>
  <wsnt:InitialTerminationTime>PT1M</wsnt:InitialTerminationTime>
</wsnt:Subscribe>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response to events.Subscribe

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:SOAP-ENC="http://www.w3.org/2003/05/soap-encoding"
xmlns:wsa5="http://www.w3.org/2005/08/addressing"
xmlns:wsnt="http://docs.oasis-open.org/wsn/b-2">
  <SOAP-ENV:Header>
    <wsa5:Action SOAP-ENV:mustUnderstand="true">http://docs.oasis-open.org/wsn/bw-
2/NotificationProducer/SubscribeResponse</wsa5:Action>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <wsnt:SubscribeResponse>
      <wsnt:SubscriptionReference>
        <wsa5:Address>http://192.168.1.16/onvif/services</wsa5:Address>
```

```
        <wsa5:ReferenceParameters>
          <dom0:SubscriptionId
xmlns:dom0="http://www.example.com/2009/event">14</dom0:SubscriptionId>
        </wsa5:ReferenceParameters>
      </wsnt:SubscriptionReference>
      <wsnt:CurrentTime>2010-12-22T17:25:48Z</wsnt:CurrentTime>
      <wsnt:TerminationTime>2010-12-22T17:26:48Z</wsnt:TerminationTime>
    </wsnt:SubscribeResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.7 SOAP Communication Traces for Storage

B.7.1 SOAP Communication Traces for Finding a Recording

The following traces refer to Section 10.1.

B.7.1.1 GetAudioOutputs

Request deviceIO:GetAudioOutputs

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
  <SOAP-ENV:Body>
     <trt:GetAudioOutputs xmlns:trt="http://www.onvif.org/ver10/media/wsdl"/>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response deviceIO:GetAudioOutputsResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
  <SOAP-ENV:Body>
    <trt:GetAudioOutputsResponse xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
      <trt:AudioOutputs token="AudioOut"/>
      </trt:GetAudioOutputsResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.7.1.2 GetAudioOutputConfiguration

Request deviceIO:GetAudioOutputConfiguration

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
  <SOAP-ENV:Body>
    <tmd:GetAudioOutputConfiguration
xmlns:tmd="http://www.onvif.org/onvif/ver10/deviceIO/wsdl">
      <tmd:AudioOutputToken>AudioOut</tmd:AudioOutputToken>
    </tmd:GetAudioOutputConfiguration>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response deviceIO:GetAudioOutputConfigurationResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema">
  <SOAP-ENV:Body>
    <tmd:GetAudioOutputConfigurationResponse
xmlns:tmd="http://www.onvif.org/onvif/ver10/deviceIO/wsdl">
      <tmd:AudioOutputConfiguration token="AudioOutCfg">
        <tt:Name>MyAudioOutCfg</tt:Name>
        <tt:UseCount>0</tt:UseCount>
        <tt:OutputToken>AudioOut</tt:OutputToken>
        <tt:OutputLevel>60</tt:OutputLevel>
        </tm:dAudioOutputConfiguration>
      </tmd:GetAudioOutputConfigurationResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.7.1.3 GetAudioDecoderConfiguration

Request media:GetAudioDecoderConfigurationOptions

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
  <SOAP-ENV:Body>
    <trt:GetAudioDecoderConfigurationOptions
xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
    </trt:GetAudioDecoderConfigurationOptions>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response media:GetAudioDecoderConfigurationOptionsResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-
envelope"xmlns:tt="http://www.onvif.org/ver10/schema">
  <SOAP-ENV:Body>
    <trt:GetAudioDecoderConfigurationOptionsResponse
xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
      <trt:Options>
        <tt:AACDecOptions>
          <tt:Bitrate>
            <tt:Items>96</tt:Items>
          </tt:Bitrate>
          <tt:SampleRateRange>
            <tt:Items>16</tt:Items>
          </tt:SampleRateRange>
        </tt:AACDecOptions>
        <tt:G711DecOptions>
          <tt:Bitrate>
            <tt:Items>64</tt:Items>
          </tt:Bitrate>
          <tt:SampleRateRange>
            <tt:Items>8</tt:Items>
          </tt:SampleRateRange>
        </tt:G711DecOptions>
      </trt:Options>
    </trt:GetAudioDecoderConfigurationOptionsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.7.1.4 AddAudioOutputConfiguration

Request media:AddAudioOutputConfiguration

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope" >
  <SOAP-ENV:Body>
    <trt:AddAudioOutputConfiguration
xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
      <trt:ProfileToken>Profile0</trt:ProfileToken>
      <trt:ConfigurationToken>AudioOutCfg</trt:ConfigurationToken>
    </trt:AddAudioOutputConfiguration>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response media:AddAudioOutputConfigurationResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
  <SOAP-ENV:Body>
    <trt:AddAudioOutputConfigurationResponse
xmlns:trt="http://www.onvif.org/ver10/media/wsdl"/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.7.1.5 AddAudioDecoderConfiguration

Request media:AddAudioDecoderConfiguration

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
  <SOAP-ENV:Body>
    <trt:AddAudioDecoderConfiguration
xmlns:trt="http://www.onvif.org/ver10/media/wsdl">
      <trt:ProfileToken>Profile0</trt:ProfileToken>
      <trt:ConfigurationToken>AudioDecCfg</trt:ConfigurationToken>
    </trt:AddAudioDecoderConfiguration>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response media:AddAudioDecoderConfigurationResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
  <SOAP-ENV:Body>
    <trt:AddAudioDecoderConfigurationResponse
xmlns:trt="http://www.onvif.org/ver10/media/wsdl"/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.7.1.6 GetRecordings

Request recording:GetRecordings

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
  <SOAP-ENV:Body>
    <trc:GetRecordings xmlns:trc="http://www.onvif.org/recording/wsdl"/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response recording:GetRecordingsResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema">
  <SOAP-ENV:Body>
    <trc:GetRecordingsResponse xmlns:trc="http://www.onvif.org/recording/wsdl">
      <trc:RecordingItem>
        <tt:RecordingToken>Rec0</tt:RecordingToken>
```

```
        <tt:Configuration>
          <tt:Source>
            <tt:SourceId>SourceID</tt:SourceId>
            <tt:Name>Building 7</tt:Name>
            <tt:Location>Room 3</tt:Location>
            <tt:Description>Camera 45</tt:Description></tt:Description>
            <tt:Address>http://160.10.64.10/onvif/media</tt:Address>
          </tt:Source>
          <tt:Content>Video from Camera 45</tt:Content>
          <tt:MaximumRetentionTime>PT15M</tt:MaximumRetentionTime>
        </tt:Configuration>
        <tt:Tracks>
          <tt:Track>
            <tt:TrackToken>Track1</tt:TrackToken>
            <tt:Configuration>
              <tt:TrackType>Video</tt:TrackType>
              <tt:Description>VideoTrack</tt:Description>
            </tt:Configuration>
          </tt:Track>
        </tt:Tracks>
      </trc:RecordingItem>
    </trc:GetRecordingsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.7.1.7 SetRecordingConfiguration

Request recording:SetRecordingConfiguration

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema">
  <SOAP-ENV:Body>
    <trc:SetRecordingConfiguration xmlns:trc="http://www.onvif.org/recording/wsdl">
      <trc:RecordingToken>Rec0</trc:RecordingToken>
      <trc:RecordingConfiguration>
        <tt:Source>
          <tt:SourceId>Device 1</tt:SourceId>
          <tt:Name>camera PT677X</tt:Name>
          <tt:Location>Room1</tt:Location>
          <tt:Description>continuous recording of room 1</tt:Description>
          <tt:Address>192.168.0.2</tt:Address>
        </tt:Source>
        <tt:Content>Recording from device 1</tt:Content>
        <tt:MaximumRetentionTime>PT0S</tt:MaximumRetentionTime>
      </trc:RecordingConfiguration>
    </trc:SetRecordingConfiguration>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response recording:SetRecordingConfigurationResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
  <SOAP-ENV:Body>
    <trc:SetRecordingConfigurationResponse
xmlns:trc="http://www.onvif.org/recording/wsdl"/>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.7.1.8 CreateRecordingJob

Request recording:CreateRecordingJob

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema">
  <SOAP-ENV:Body>
    <trc:CreateRecordingJob xmlns:trc="http://www.onvif.org/recording/wsdl">
      <trc:JobConfiguration>
        <tt:RecordingToken>Rec0</tt:RecordingToken>
        <tt:Mode>Active</tt:Mode>
        <tt:Priority>1</tt:Priority>
        <tt:Source>
          <tt:SourceToken Type="http://www.onvif.org/ver10/schema/Profile">
            <tt:Token>Profile1</tt:Token>
          </tt:SourceToken>
          <tt:AutoCreateReceiver>false</tt:AutoCreateReceiver>
        </tt:Source>
      </trc:JobConfiguration>
    </trc:CreateRecordingJob>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response recording:CreateRecordingJobResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema">
  <SOAP-ENV:Body>
    <trc:CreateRecordingJobResponse xmlns:trc="http://www.onvif.org/recording/wsdl">
      <trc:JobToken>RecJob</trc:JobToken>
      <trc:JobConfiguration>
        <tt:RecordingToken>Rec0</tt:RecordingToken>
        <tt:Mode>Active</tt:Mode>
        <tt:Priority>1</tt:Priority>
        <tt:Source>
          <tt:SourceToken Type="http://www.onvif.org/ver10/schema/Profile">
            <tt:Token>Profile1</tt:Token>
          </tt:SourceToken>
          <tt:Tracks>
            <tt:SourceTag>Video</tt:SourceTag>
            <tt:Destination>Video Track</tt:Destination>
          </tt:Tracks>
        </tt:Source>
      </trc:JobConfiguration>
    </trc:CreateRecordingJobResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.7.1.9 FindEvents

Request search:FindEvents

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema" xmlns:wsnt="http://docs.oasis-
open.org/wsn/b-2" xmlns:tns1="http://www.onvif.org/ver10/topics">
  <SOAP-ENV:Body>
    <tse:FindEvents xmlns:tse="http://www.onvif.org/search/wsdl">
      <tse:StartPoint>2010-12-24T08:00:00.0Z</tse:StartPoint>
      <tse:EndPoint>2001-12-17T09:30:47.0Z</tse:EndPoint>
      <tse:Scope>
```

```
        <tt:IncludedRecordings>MyRec</tt:IncludedRecordings>
      </tse:Scope>
      <tse:SearchFilter>
        <wsnt:TopicExpression
Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
            tns1:RecordingHistory/Track/State
        </wsnt:TopicExpression>
      </tse:SearchFilter>
      <tse:IncludeStartState>false</tse:IncludeStartState>
      <tse:MaxMatches>100</tse:MaxMatches>
      <tse:KeepAliveTime>PT1M</tse:KeepAliveTime>
    </tse:FindEvents>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response search:FindEventsResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
  <SOAP-ENV:Body>
    <tse:FindEventsResponse xmlns:tse="http://www.onvif.org/search/wsdl">
      <tse:SearchToken>MySearchToken</tse:SearchToken>
    </tse:FindEventsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.7.1.10 GetRecordingSearchResults

Request search:GetRecordingSearchResult

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope">
  <SOAP-ENV:Body>
    <tse:GetRecordingSearchResults xmlns:tse="http://www.onvif.org/search/wsdl">
      <tse:SearchToken>MySearchToken</tse:SearchToken>
      <tse:WaitTime>PT1M</tse:WaitTime>
    </tse:GetRecordingSearchResults>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response search:GetRecordingSearchResultResponse

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2003/05/soap-envelope"
xmlns:tt="http://www.onvif.org/ver10/schema" xmlns:wsnt="http://docs.oasis-
open.org/wsn/b-2" xmlns:tns1="http://www.onvif.org/ver10/topics">
  <SOAP-ENV:Body>
    <tse:GetEventSearchResultsResponse xmlns:tse="http://www.onvif.org/search/wsdl">
      <tse:ResultList>
        <tt:SearchState>Completed</tt:SearchState>
        <tt:Result>
          <tt:RecordingToken>MyRec</tt:RecordingToken>
          <tt:TrackToken>VideoTrack</tt:TrackToken>
          <tt:Time>2010-12-24T09:30:47.0Z</tt:Time>
          <tt:Event>
            <wsnt:Topic
Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
              tns1:RecordingHistory/Track/State
            </wsnt:Topic>
            <wsnt:Message>
              <tt:Message>
```

```
                <tt:Source>
                  <tt:SimpleItem Name="RecordingToken" Value="MyRecording"/>
                  <tt:SimpleItem Name="Track" Value="VideoTrack"/>
                </tt:Source>
                <tt:Data>
                  <tt:SimpleItem Name="IsDataPresent" Value="true"/>
                </tt:Data>
              </tt:Message>
            </wsnt:Message>
          </tt:Event>
          <tt:StartStateEvent>false</tt:StartStateEvent>
        </tt:Result>
        <tt:Result>
          <tt:RecordingToken>MyRec</tt:RecordingToken>
          <tt:TrackToken>VideoTrack</tt:TrackToken>
          <tt:Time>2010-12-24T09:30:48.0Z</tt:Time>
          <tt:Event>
            <wsnt:Topic
Dialect="http://www.onvif.org/ver10/tev/topicExpression/ConcreteSet">
              tns1:RecordingHistory/Track/State
            </wsnt:Topic>
            <wsnt:Message>
              <tt:Message>
                <tt:Source>
                  <tt:SimpleItem Name="RecordingToken" Value="MyRecording"/>
                  <tt:SimpleItem Name="Track" Value="VideoTrack"/>
                </tt:Source>
                <tt:Data>
                  <tt:SimpleItem Name="IsDataPresent" Value="false"/>
                </tt:Data>
              </tt:Message>
            </wsnt:Message>
          </tt:Event>
          <tt:StartStateEvent>false</tt:StartStateEvent>
        </tt:Result>
      </tse:ResultList>
    </tse:GetEventSearchResultsResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

B.8 SOAP Communication Traces for Display

B.8.1 SOAP Communication Traces for Configuring a Display Device to Show a Stream

The following traces refer to Section 11.1.

B.8.1.1 GetVideoOutputs

SOAP REQUEST

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
   xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
   xmlns:tmd="http://www.onvif.org/onvif/ver10/deviceIO/wsdl"
>

<soap:Body>
      <tmd:GetVideoOutputs/>
   </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd="http://www.onvif.org/onvif/ver10/deviceIO/wsdl"
  xmlns:tt="http://www.onvif.org/ver10/schema"
>

<soap:Body>
      <tmd:GetVideoOutputsResponse>
         <tmd:VideoOutputs token="VideoOutputToken0">
            <tt:Layout>
               <tt:PaneLayout>
                  <tt:Pane>PaneToken0</tt:Pane>
                  <tt:Area bottom="-1.0" top="1.0" right="1.0" left="-1.0"/>
               </tt:PaneLayout>
            </tt:Layout>
         </tmd:VideoOutputs>
      </tmd:GetVideoOutputsResponse>
   </soap:Body>
</soap:Envelope>
```

B.8.1.2 GetReceivers

SOAP REQUEST

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd="http://www.onvif.org/onvif/ver10/receiver/wsdl"
>

<soap:Body>
      <tmd:GetReceivers/>
   </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:tmd="http://www.onvif.org/onvif/ver10/receiver/wsdl"
xmlns:tt="http://www.onvif.org/ver10/schema">

<soap:Body>
      <tmd:GetReceiversResponse>
         <tmd:Receivers>
            <tt:Token>ReceiverToken0</tt:Token>
            <tt:Configuration>
               <tt:Mode>NeverConnect</tt:Mode>
               <tt:MediaUri>rstp://camera-device/stream/live</tt:MediaUri>
               <tt:StreamSetup>
                  <tt:Stream>RTP-Unicast</tt:Stream>
                  <tt:Transport>
                     <tt:Protocol>UDP</tt:Protocol>
                  </tt:Transport>
               </tt:StreamSetup>
            </tt:Configuration>
         </tmd:Receivers>
      </tmd:GetReceiversResponse>
   </soap:Body>
</soap:Envelope>
```

B.8.1.3 GetPaneConfiguration

SOAP REQUEST

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd="http://www.onvif.org/onvif/ver10/display/wsdl"
>

<soap:Body>
      <tmd:GetPaneConfiguration>
         <tmd:VideoOutput>VideoOut0</tmd:VideoOutput>
         <tmd:Pane>PaneConfig0</tmd:Pane>
      </tmd:GetPaneConfiguration>
   </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd="http://www.onvif.org/onvif/ver10/display/wsdl"
  xmlns:tt="http://www.onvif.org/ver10/schema"
>

<soap:Body>
      <tmd:GetPaneConfigurationResponse>
         <tmd:PaneConfiguration>
            <tt:PaneName>PaneName0</tt:PaneName>
            <tt:Token>PaneToken0</tt:Token>
         </tmd:PaneConfiguration>
      </tmd:GetPaneConfigurationResponse>
```

```
   </soap:Body>
</soap:Envelope>
```

B.8.1.4 SetPaneConfiguration

SOAP REQUEST

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd="http://www.onvif.org/onvif/ver10/display/wsdl"
  xmlns:tt="http://www.onvif.org/ver10/schema"
>

<soap:Body>
      <tmd:SetPaneConfiguration>
         <tmd:VideoOutput>VideoOutput0</tmd:VideoOutput>
         <tmd:PaneConfiguration>
            <tt:PaneName>PaneName0</tt:PaneName>
            <tt:ReceiverToken>ReceiverToken0</tt:ReceiverToken>
            <tt:Token>PaneToken0</tt:Token>
         </tmd:PaneConfiguration>
      </tmd:SetPaneConfiguration>
   </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd="http://www.onvif.org/onvif/ver10/display/wsdl"
>

<soap:Body>
      <tmd:SetPaneConfigurationResponse/>
   </soap:Body>
</soap:Envelope>
```

B.8.1.5 SetReceiverMode

SOAP REQUEST

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd="http://www.onvif.org/onvif/ver10/display/wsdl"
>

<soap:Body>
   <soap:Body>
      <tmd:SetReceiverMode>
         <tmd:ReceiverToken>ReceiverToken0</tmd:ReceiverToken>
         <tmd:Mode>AlwaysConnect</tmd:Mode>
      </tmd:SetReceiverMode>
   </soap:Body>
  </soap:Body>
</soap:Envelope>
```

SOAP RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:tmd="http://www.onvif.org/onvif/ver10/receiver/wsdl">

<soap:Body>
      <tmd:SetReceiverModeResponse/>
   </soap:Body>
</soap:Envelope>
```

B.8.2 SOAP Communication Traces for Creating and Deleting PaneConfiguration

The following traces refer to Section 11.2.

B.8.2.1 GetLayout

REQUEST

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:tmd=„http://www.onvif.org/onvif/ver10/display/wsdl“
>

<soap:Body>
      <tmd:GetLayout>
         <tmd:VideoOutput>VideoOutputToken0</tmd:VideoOutput>
      </tmd:GetLayout>
   </soap:Body>
</soap:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd=„http://www.onvif.org/onvif/ver10/display/wsdl“
  xmlns:tt="http://www.onvif.org/ver10/schema"
>

<soap:Body>
      <tmd:GetLayoutResponse>
         <tmd:Layout>
            <tt:PaneLayout>
            <tt:Layout>
               <tt:Pane>PaneToken0</tt:Pane>
               <tt:Area bottom="-1.0" top="0.0" right="0.0" left="-1.0"/>
            </tt:PaneLayout>
         </tmd:Layout>
      </tmd:GetLayoutResponse>
   </soap:Body>
</soap:Envelope>
```

B.8.2.2 GetDisplayOptions

REQUEST

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:tmd=„http://www.onvif.org/onvif/ver10/display/wsdl“
>

<soap:Body>
      <tmd:GetDisplayOptions>
```

```
         <tmd:VideoOutput>VideoOutputToken0</tmd:VideoOutput>
      </tmd:GetDisplayOptions>
   </soap:Body>
</soap:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:tmd=„http://www.onvif.org/onvif/ver10/display/wsdl“
xmlns:tt="http://www.onvif.org/ver10/schema">

<soap:Body>
      <tmd:GetDisplayOptionsResponse>
         <tmd:CodingCapabilities>
            <tt:VideoDecodingCapabilities>
               <tt:JpegDecOptions>
                  <tt:ResolutionsAvailable>
                     <tt:Width>320</tt:Width>
                     <tt:Height>420</tt:Height>
                  </tt:ResolutionsAvailable>
                  <tt:SupportedInputBitrate>
                     <tt:Min>1</tt:Min>
                     <tt:Max>8000</tt:Max>
                  </tt:SupportedInputBitrate>
                  <tt:SupportedFrameRate>
                     <tt:Min>1</tt:Min>
                     <tt:Max>30</tt:Max>
                  </tt:SupportedFrameRate>
               </tt:JpegDecOptions>
            </tt:VideoDecodingCapabilities>
         </tmd:CodingCapabilities>
      </tmd:GetDisplayOptionsResponse>
   </soap:Body>
</soap:Envelope>
```

B.8.2.3 CreatePaneConfiguration

REQUEST

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd=„http://www.onvif.org/onvif/ver10/display/wsdl“
  xmlns:tt="http://www.onvif.org/ver10/schema"
>

<soap:Body>
      <tmd:CreatePaneConfiguration>
         <tmd:VideoOutput>VideoOutputToken0</tmd:VideoOutput>
         <tmd:PaneConfiguration>
            <tt:PaneName>PaneName0</tt:PaneName>
            <tt:ReceiverToken>ReveiverToken0</tt:ReceiverToken>
            <tt:Token>PaneToken0</tt:Token>
         </tmd:PaneConfiguration>
      </tmd:CreatePaneConfiguration>
```

```
   </soap:Body>
</soap:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:tmd=„http://www.onvif.org/onvif/ver10/display/wsdl“>

<soap:Body>
      <tmd:CreatePaneConfigurationResponse/>
   </soap:Body>
</soap:Envelope>
```

B.8.2.4 SetLayout

REQUEST

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd=„http://www.onvif.org/onvif/ver10/display/wsdl“
  xmlns:tt="http://www.onvif.org/ver10/schema"
>

<soap:Body>
      <tmd:SetLayout>
         <tmd:VideoOutput>?</tmd:VideoOutput>
         <tmd:Layout>
            <tt:PaneLayout>
               <tt:Pane>PaneToken0</tt:Pane>
               <tt:Area bottom="-1.0" top="0.0" right="0.0" left="-1.0"/>
            </tt:PaneLayout>
            <tt:PaneLayout>
               <tt:Pane>PaneToken1</tt:Pane>
               <tt:Area bottom="0.0" top="1.0" right="1.0" left="0.0"/>
            </tt:PaneLayout>
         </tmd:Layout>
      </tmd:SetLayout>
   </soap:Body>
</soap:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd=„http://www.onvif.org/onvif/ver10/display/wsdl“
>

<soap:Body>
      <tmd:SetLayoutResponse/>
   </soap:Body>
</soap:Envelope>
```

B.8.2.5 DeletePaneConfiguration

REQUEST

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd=„http://www.onvif.org/onvif/ver10/display/wsdl“
>

<soap:Body>
      <tmd:DeletePaneConfiguration>
         <tmd:VideoOutput>VideoOut0</tmd:VideoOutput>
         <tmd:PaneToken>Pane0</tmd:PaneToken>
      </tmd:DeletePaneConfiguration>
   </soap:Body>
</soap:Envelope>
```

RESPONSE – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd=„http://www.onvif.org/onvif/ver10/display/wsdl“
>
   <soap:Body>
      <tmd:DeletePaneConfiguration/>
   </soap:Body>
</soap:Envelope>
```

B.8.3 SOAP Communication Traces for Changing the Layout Based on LayoutOptions

B.8.3.1 GetDisplayOptions

For this request, see Section 11.3.

The following response provides `LayoutOptions` for a device that can do two types of layout on the video output: 1x1 and 2x2 panes.

Response – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:tmd=„http://www.onvif.org/onvif/ver10/display/wsdl“
xmlns:tt="http://www.onvif.org/ver10/schema">
   <soap:Body>
      <tmd:GetDisplayOptionsResponse>
         <tmd:LayoutOptions>
            <tt:PaneLayoutOptions>
               <tt:Area bottom="-1.0" top="1.0" right="1.0" left="-1.0"/>
            </tt:PaneLayoutOptions>
            <tt:PaneLayoutOptions>
               <tt:Area bottom="0.0" top="1.0" right="0.0" left="-1.0"/>
               <tt:Area bottom="0.0" top="1.0" right="1.0" left="0.0"/>
               <tt:Area bottom="-1.0" top="0.0" right="0.0" left="-1.0"/>
               <tt:Area bottom="-1.0" top="0.0" right="1.0" left="0.0"/>
            </tt:PaneLayoutOptions>
         </tmd:LayoutOptions>
         <tmd:CodingCapabilities>
            <tt:VideoDecodingCapabilities>
```

```
                <tt:JpegDecOptions>
                   <tt:ResolutionsAvailable>
                      <tt:Width>320</tt:Width>
                      <tt:Height>420</tt:Height>
                   </tt:ResolutionsAvailable>
                   <tt:SupportedInputBitrate>
                      <tt:Min>1</tt:Min>
                      <tt:Max>8000</tt:Max>
                   </tt:SupportedInputBitrate>
                   <tt:SupportedFrameRate>
                      <tt:Min>1</tt:Min>
                      <tt:Max>30</tt:Max>
                   </tt:SupportedFrameRate>
                </tt:JpegDecOptions>
             </tt:VideoDecodingCapabilities>
          </tmd:CodingCapabilities>
       </tmd:GetDisplayOptionsResponse>
    </soap:Body>
</soap:Envelope>
```

B.8.3.2 SetLayout

For this example, the 2x2 layout is used.

Request display:**SetLayout**

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd=„http://www.onvif.org/onvif/ver10/display/wsdl“
  xmlns:tt="http://www.onvif.org/ver10/schema"
>
   <soap:Body>
      <tmd:SetLayout>
         <tmd:VideoOutput>?</tmd:VideoOutput>
         <tmd:Layout>
            <tt:PaneLayout>
               <tt:Pane>PaneToken0</tt:Pane>
               <tt:Area bottom="0.0" top="1.0" right="0.0" left="-1.0" />
            </tt:PaneLayout>
            <tt:PaneLayout>
               <tt:Pane>PaneToken1</tt:Pane>
               <tt:Area bottom="0.0" top="1.0" right="1.0" left="0.0"/>
            </tt:PaneLayout>
            <tt:PaneLayout>
               <tt:Pane>PaneToken2</tt:Pane>
               <tt:Area bottom="-1.0" top="0.0" right="0.0" left="-1.0"/>
            </tt:PaneLayout>
            <tt:PaneLayout>
               <tt:Pane>PaneToken3</tt:Pane>
               <tt:Area bottom="-1.0" top="0.0" right="1.0" left="0.0"/>
            </tt:PaneLayout>
         </tmd:Layout>
      </tmd:SetLayout>
   </soap:Body>
</soap:Envelope>
```

NOTICE: The order of the areas reflects the order provided by the `LayoutOptions` element from the trace example in section B.8.3.1. This order is highly recommended, because some display device implementations might insist and depend on this strict ordering.

B.8.4 SOAP Communication Traces for Configuring a Receiver Based on DecoderCapabilities

The following traces refer to Section 11.4.

B.8.4.1 GetReceivers

Request receiver:**GetReceivers**

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:tmd="http://www.onvif.org/ver10/receiver/wsdl"
>
   <soap:Body>
      <tmd:GetReceivers/>
   </soap:Body>
</soap:Envelope>
```

Response – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
xmlns:tmd="http://www.onvif.org/ver10/receiver/wsdl"
xmlns:tt="http://www.onvif.org/ver10/schema"
>
   <soap:Body>
      <tmd:GetReceiversResponse>
         <!--Zero or more repetitions:-->
         <tmd:Receivers>
            <tt:Token>ReceiverToken0</tt:Token>
            <tt:Configuration>
               <tt:Mode>NeverConnect</tt:Mode>
               <tt:MediaUri>rtsp://1.2.3.4/media/live</tt:MediaUri>
               <tt:StreamSetup>
                  <tt:Stream>UDP-Unicast</tt:Stream>
                  <tt:Transport>
                     <tt:Protocol>RTSP</tt:Protocol>
                  </tt:Transport>
               </tt:StreamSetup>
            </tt:Configuration>
         </tmd:Receivers>
         <tmd:Receivers>
            <tt:Token>ReceiverToken1</tt:Token>
            <tt:Configuration>
               <tt:Mode>NeverConnect</tt:Mode>
               <tt:MediaUri>rtsp://2.3.4.5/</tt:MediaUri>
               <tt:StreamSetup>
                  <tt:Stream>UDP-Unicast</tt:Stream>
                  <tt:Transport>
                     <tt:Protocol>HTTP</tt:Protocol>
                  </tt:Transport>
               </tt:StreamSetup>
            </tt:Configuration>
         </tmd:Receivers>
      </tmd:GetReceiversResponse>
   </soap:Body>
</soap:Envelope>
```

This device holds two receiver instances.

B.8.4.2 CreateReceiver

Request receiver:**CreateReceiver**

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd="http://www.onvif.org/ver10/receiver/wsdl"
  xmlns:tt="http://www.onvif.org/ver10/schema"
>
   <soap:Body>
      <tmd:CreateReceiver>
         <tmd:Configuration>
            <tt:Mode>NeverConnect</tt:Mode>
            <tt:MediaUri>http://4.5.6.7/media/live</tt:MediaUri>
            <tt:StreamSetup>
               <tt:Stream>UDP-Unicast</tt:Stream>
               <tt:Transport>
                  <tt:Protocol>HTTP</tt:Protocol>
              </tt:Transport>
            </tt:StreamSetup>
         </tmd:Configuration>
      </tmd:CreateReceiver>
   </soap:Body>
</soap:Envelope>
```

Response – on success

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:tmd="http://www.onvif.org/ver10/receiver/wsdl"
  xmlns:tt="http://www.onvif.org/ver10/schema"
>
   <soap:Body>
      <tmd:CreateReceiverResponse>
         <tmd:Receiver>
            <tt:Token>ReceoverToken3</tt:Token>
            <tmd:Configuration>
                <tt:Mode>NeverConnect</tt:Mode>
                <tt:MediaUri>http://4.5.6.7/media/live</tt:MediaUri>
                <tt:StreamSetup>
                   <tt:Stream>UDP-Unicast</tt:Stream>
                   <tt:Transport>
                      <tt:Protocol>HTTP</tt:Protocol>
                  </tt:Transport>
               </tt:StreamSetup>
            </tmd:Configuration>
         </tmd:Receiver>
      </tmd:CreateReceiverResponse>
   </soap:Body>
</soap:Envelope>
```

Annex C List of Functions with References

This annex provides a reference index of ONVIF functions that are described in the Application Programmers Guide. (Not all ONVIF functions are described in the *Application Programmer's Guide*, but they are included here for the sake of completeness.)

For more information on the services to which these functions belong, refer to the ONVIF 2.0 Service Operation Index at http://www.onvif.org/onvif/ver20/util/operationIndex.html.

F

G

N

O

P

Q

R

S

Annex D Pseudo Code Conventions

The information in this annex defines the basic structures that are used in this document and explains how these can be translated into real languages.

D.1 General Language Style

Each expression, if not a flow control command, must be terminated by a semicolon and placed on its own line. Each comment must be placed on a separate line that begins with two slashes to mark the start of the comment. A scope that groups expressions in a sequence must be encapsulated by curly brackets. An indention of four characters must be made within this area. The brackets shall be placed on separate lines, resulting in the following code format.

```
Object1.Command1(Parameter1);
// Comment to Object2
Object2.Command2(Parameter2);
{
    // Comment to Object3
    Object3.Command3(Parameter3);
}
```

This notation should directly map to each of the targeted real languages quite clearly.

D.2 while

If a code segment must be executed as long as a condition is met, this shall be expressed by a `while` statement. The `while` statement must always start with the evaluating `while` expression, with its condition placed inside round brackets.

```
while(Object.Condition())
{
    Object.Command();
}
```

Conditions can be all statements that evaluate to a Boolean value. This can be a variable of an according type, a function, or method returning an according type, or any correlation of variables, values and/or objects that result in a Boolean value. Correlations shall be made with mathematical expressions such as "**==**", "**!=**", "**>**", "**<**", "**>=**" or "**<=**". Negating a condition shall be done with a leading **!** symbol. Setting two or more conditions in correlation shall be done by the logic operations "&&" or "||". To get a clear defined logic, round brackets shall be used to exactly define the ordering of operations during evaluation. Exceptions to that shall be kept to a minimum.

D.3 if-else

The flow control statement for testing for a certain condition is the `if` clause. It shall encapsulate the tested expression in round brackets, and the conditional code sequence shall always be encapsulated in curly brackets. If there is a requirement to execute a command sequence if the conditional clause is not met, the `else` statement may be used without any expression. Chaining of several `if` clauses where a second expression must be evaluated only if the first condition is not met shall be allowed by using the second `if` directly following the `else` token of the first expression.

```
if (BooleanObject1)
{
    Object2.Command1();
}
else if (Object2.TestCommand())
{
    Object2.Command2();
}
```

This can be easily mapped to different languages, because most of them support this notation. In some languages, the combined `else-if` clause must be translated into a joined token like `elseif` or `elif`.

D.4 foreach

One typical scenario where flow control is required is the iteration of elements of a list. Here the `foreach` expression shall be used with two operands, an implicitly typed element variable, and a list object using the following statement.

```
foreach ( Element in ElementList )
{
    Element.Command(Parameter);
}
```

This form of flow control increases the readability and is available in most of the targeted languages.

D.5 break

The `break` statement is used to exit a looped section like `foreach` or `while`. See the following examples.

```
i=1
while( i > 0 )
{
    i=i+1;
    if ( i > 100 )
    {
        break;
    }
}

foreach ( element in list )
{
   if ( element.matches( something ) )
   {
       found=element;
       break;
   }
}
```

D.6 try catch throw

The `try catch` construct is used to handle special error cases or to do general error handling. The `try` statement marks a special guided context where exceptions are explicitly expected. All exceptions that are handled in a special way will be listed right after the `try` section in one or more separate `catch` sections marked by dedicated `catch` statements. A `catch` statement will include one or more exception types which will be separated by colons. If one or more types are listed, a variable name may follow that can be used to reference a thrown exception instance. If the `catch` statement doesn't contain any parameters, it will handle any exception unconditionally. Finally, the `throw` statement will throw an instance of an exception. A `throw` must be followed by an instantiation or a variable containing the instance to be thrown. The only exception is a `catch` context where no instance name was provided in the `catch` statement.

```
try
{
    //some code
    Throw ExceptionA();
}
catch ( ExceptionTypeA )
{
    //handling exception without using exception index
    throw;
}
catch ( ExceptionTypeB , ExceptionTypeC e )
{
    if ( e.parameter == someValue)
    {
          //special handling path one
    }
    else
    {
        throw e;
    }
}
```

D.7 optional Elements

Within the type declarations in the `onvif.xsd` document, there are some elements marked as `optional` by the statement `minOccures=0`. These optional elements can be treated in many different ways in each of the possible languages and frameworks. To increase readability, optionals are treated as a special type which includes an additional state of being not initialized or present. Therefore, a dedicated global function named `present()` is used to make that additional state accessible. To get an optional object/instance into the uninitialized state, the notation is used by assigning an uninitialized value `null`.

```
if( present(optional))
{
    Object.Command(optional);
    optional = null;
}
```

Because the implementation of optionals can differ heavily depending on the language and type of framework used to deal with SOAP and `onvif.xsd`, no straightforward mapping to any of the languages is available.