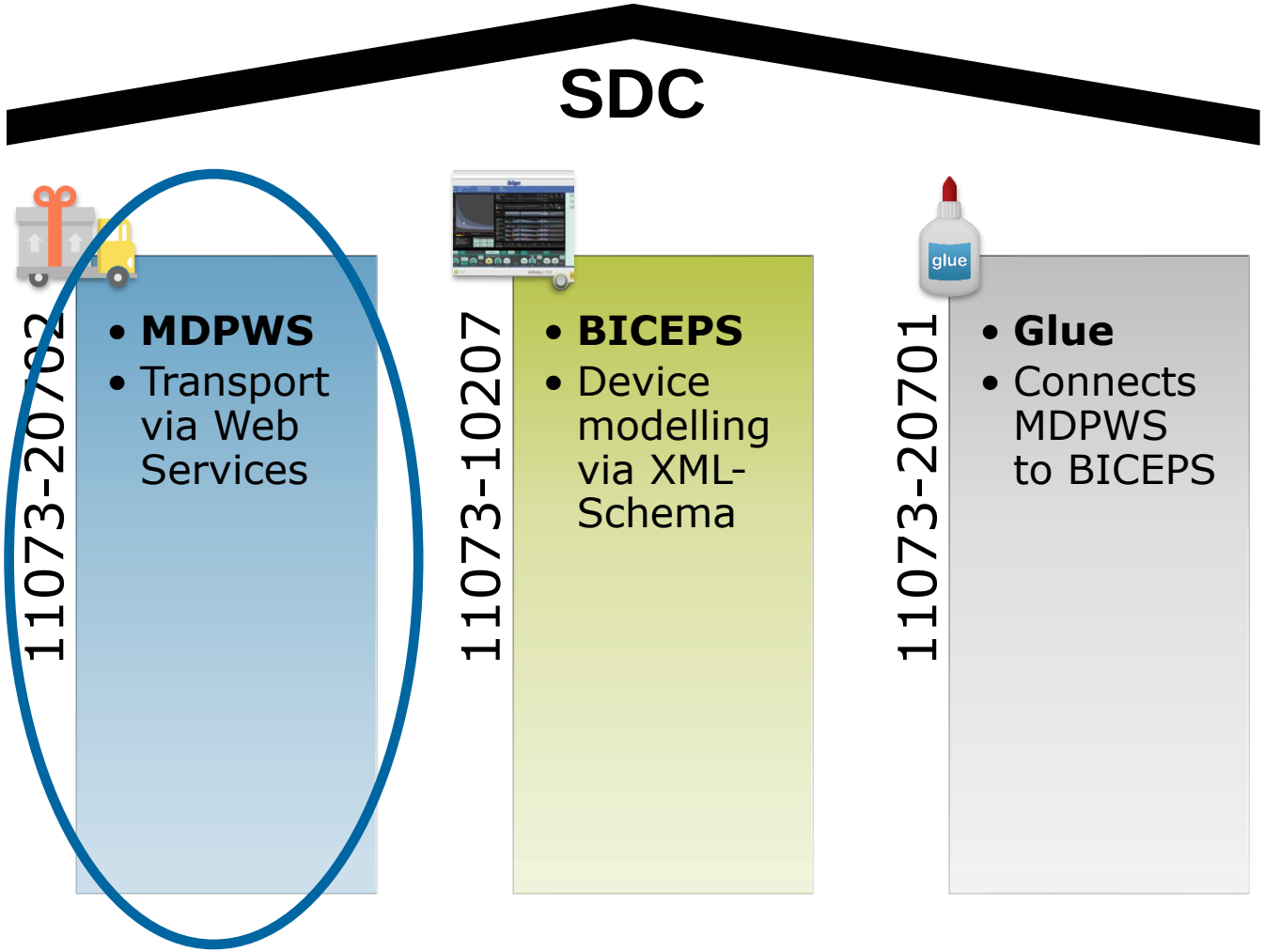
Medical Devices Profile for Web Services

Revision 1,
2018-10-01

IEEE
Advancing Technology for Humanity

Orientation



Identifying technology

Objective

- Designate a technology for IP based medical device connectivity

Requirements

- Safe and reliable data transfer
- Streaming capabilities
- Interoperability
  - Open standards
  - Widely adopted standards
- Dynamic connection establishment
  - Almost no knowledge of runtime environment
  - Plug-and-play capabilities

Identifying technology

- No middleware technology met all requirements
- However, a case study revealed that there was a candidate worth take into scope
  - *Devices Profile for Web Services*
- We decided to use that profile as a basic middleware technology and extended it to support streaming, safety capabilities, etc.
  - *Medical Devices Profile for Web Services*

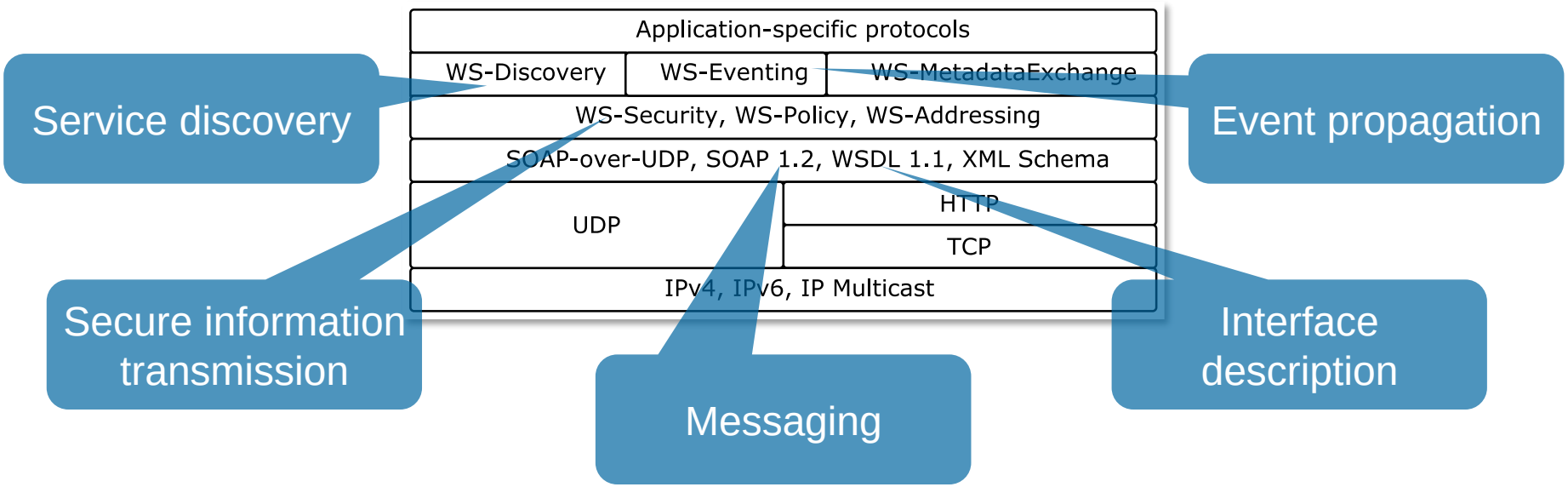
DPWS

DPWS

DPWS = Devices Profile for Web Services

- OASIS standard (07/2009)
- Utilizes a subset of WS-* standards
- Designed for resource-constrained devices

→ DPWS provides plug-and play between devices that are connected via IP networks.

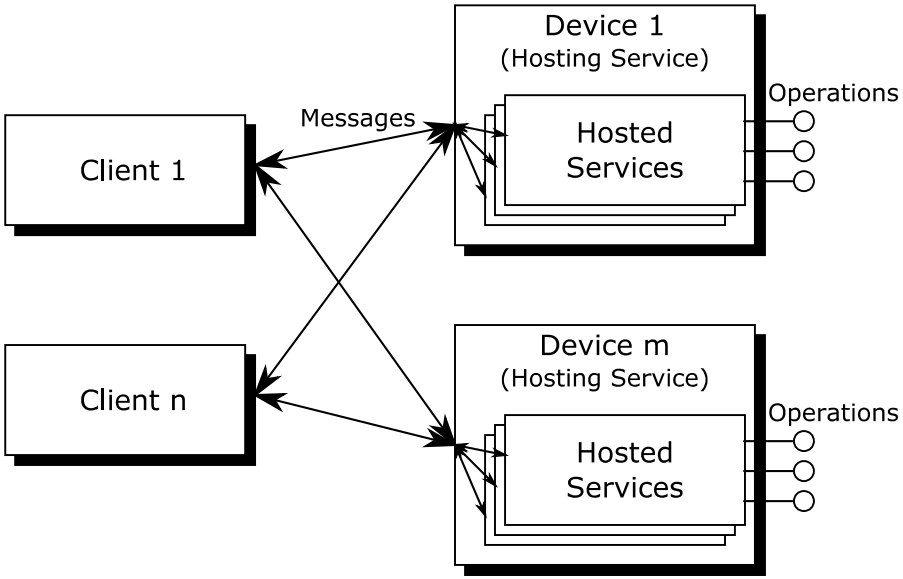


DPWS
Wording

- Device (aka Hosting Service)
  - Provides a set of Services (aka Hosted Service)
- Client
  - Requests Hosting Services to retrieve Hosted Services, on which they can invoke Operations

From a client-server perspective, the *Device* can be considered as a server, and the *Client* can be considered as a client.

However, *Device* and *Client* in terms of DPWS are able to switch roles. Hence, every *Device* can act as a client and vice versa.



DPWS
Messaging

- DPWS is based on Web Services, hence messaging is established by using SOAP documents
- DPWS supports Request-Response and Notification message exchange patterns
- A SOAP message exchanged via DPWS has the following simplified format

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
  SOAP-ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
    <SOAP-ENV:Header>
        ...
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
      ...
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

DPWS
Messaging

Moreover, DPWS makes use of WS-Addressing

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope xmlns:wsa=http://www.w3.org/2005/08/addressing ...>
  <SOAP-ENV:Header>
    <wsa:MessageID>urn:uuid:4eb70ba5-1f7a-4843-86df-977f92e8cf46</wsa:MessageID>
    <wsa:RelatesTo>urn:uuid:f3b70c20-1c5f-11e7-808d-c247d6c531f6</wsa:RelatesTo>
    <wsa:To>http://www.w3.org/2005/08/addressing/anonymous</wsa:To>
    <wsa5:Action>http://any-action-uri</wsa5:Action>
   </SOAP-ENV:Header>
...
```

DPWS
Discovery of Devices

DPWS uses WS-Discovery for implicit and explicit discovery

- Implicit discovery
  - A Device announces its presence to Clients through Hello and Bye messages
- Explicit discovery
  - A Client actively searches for Devices through Probe messages

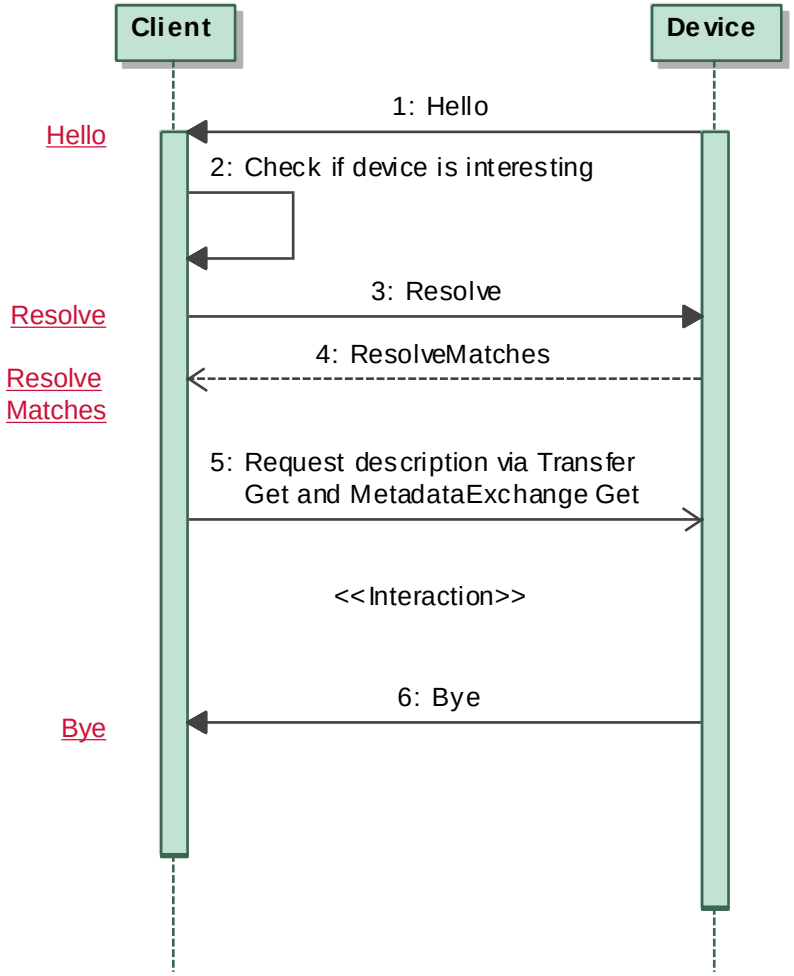
DPWS
Discovery of Devices

- Discovery is based on list of Types (QNames) and Scopes (URIs)
- WS-Discovery uses UDP, so one can use Wireshark to investigate message transfer on port 3702
  - IPv4 multicast address: 239.255.255.250
  - IPv6 multicast address: FF02::C

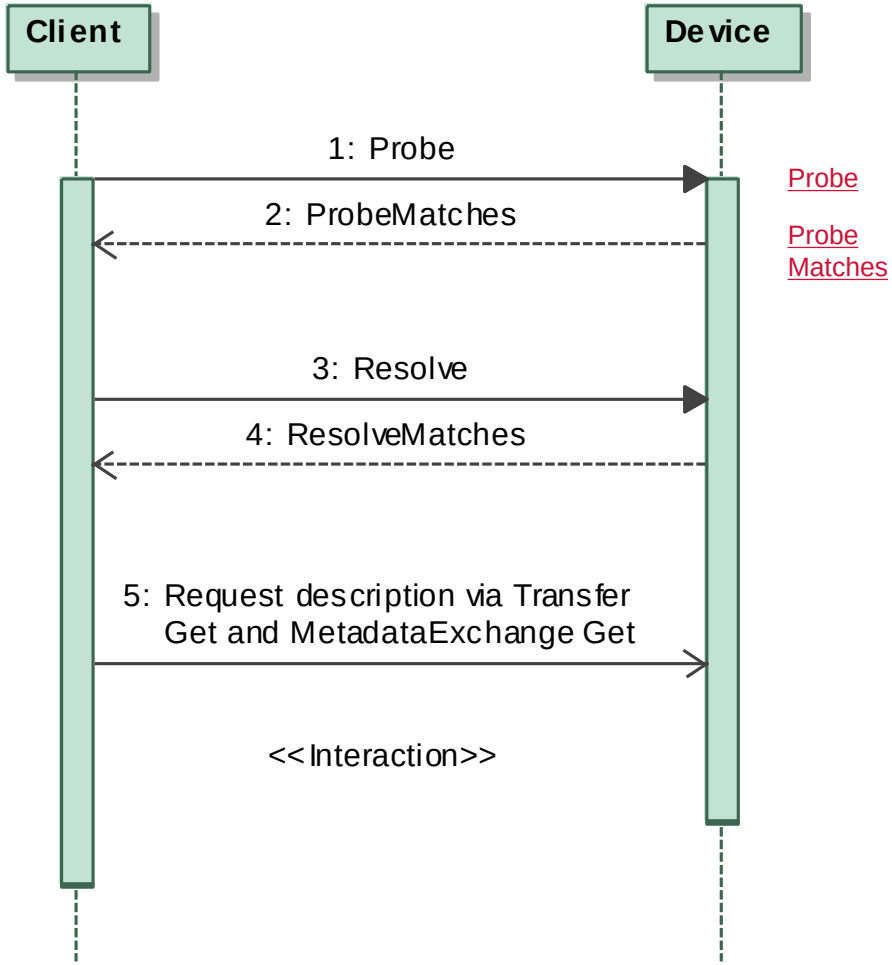
Note: In WS-Discovery speech a Client probes for Target Services. Transferred to DPWS a Target Service can be considered as the Device (aka Hosting Service).

DPWS
Discovery of Devices

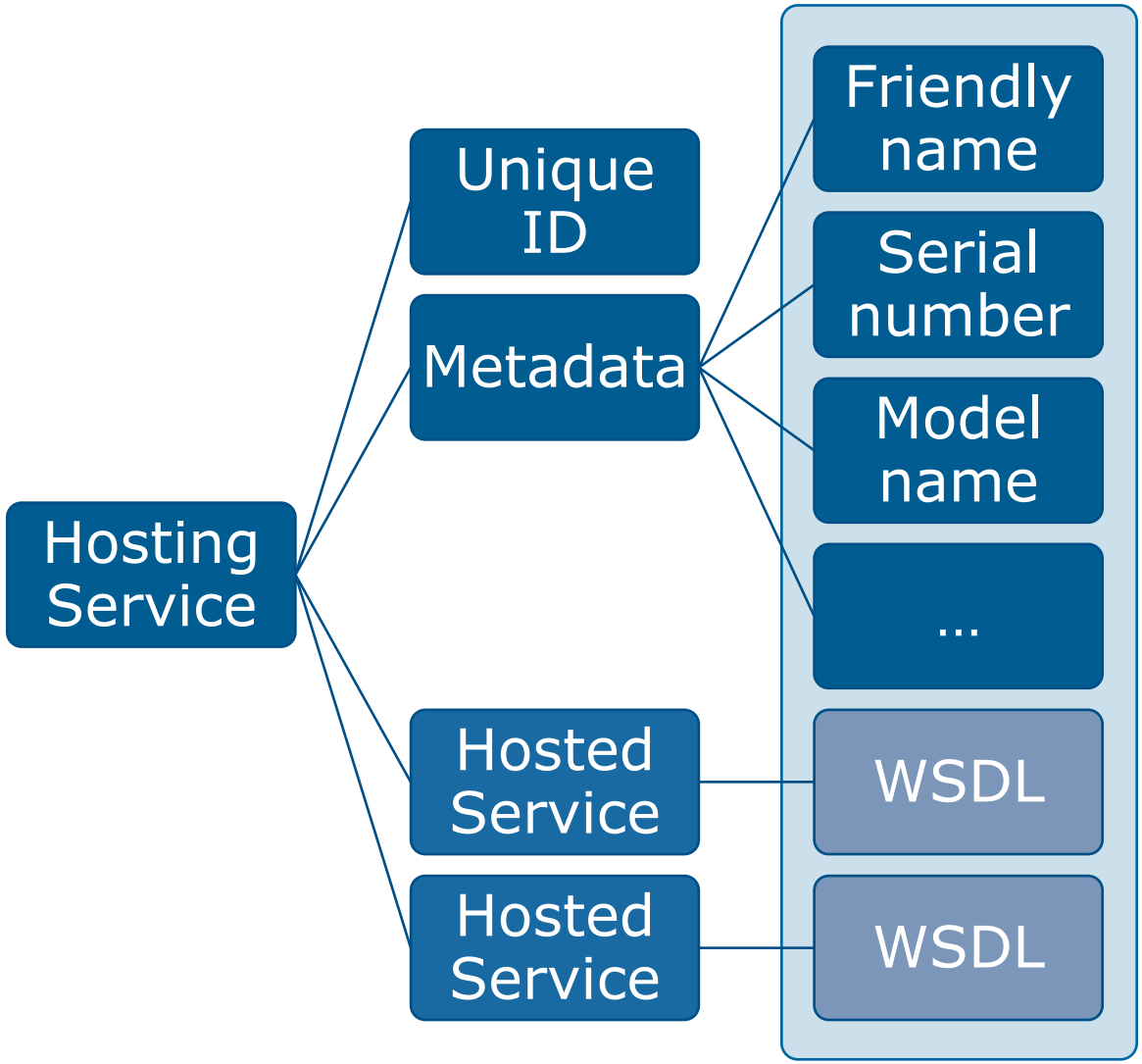
Implicit Discovery



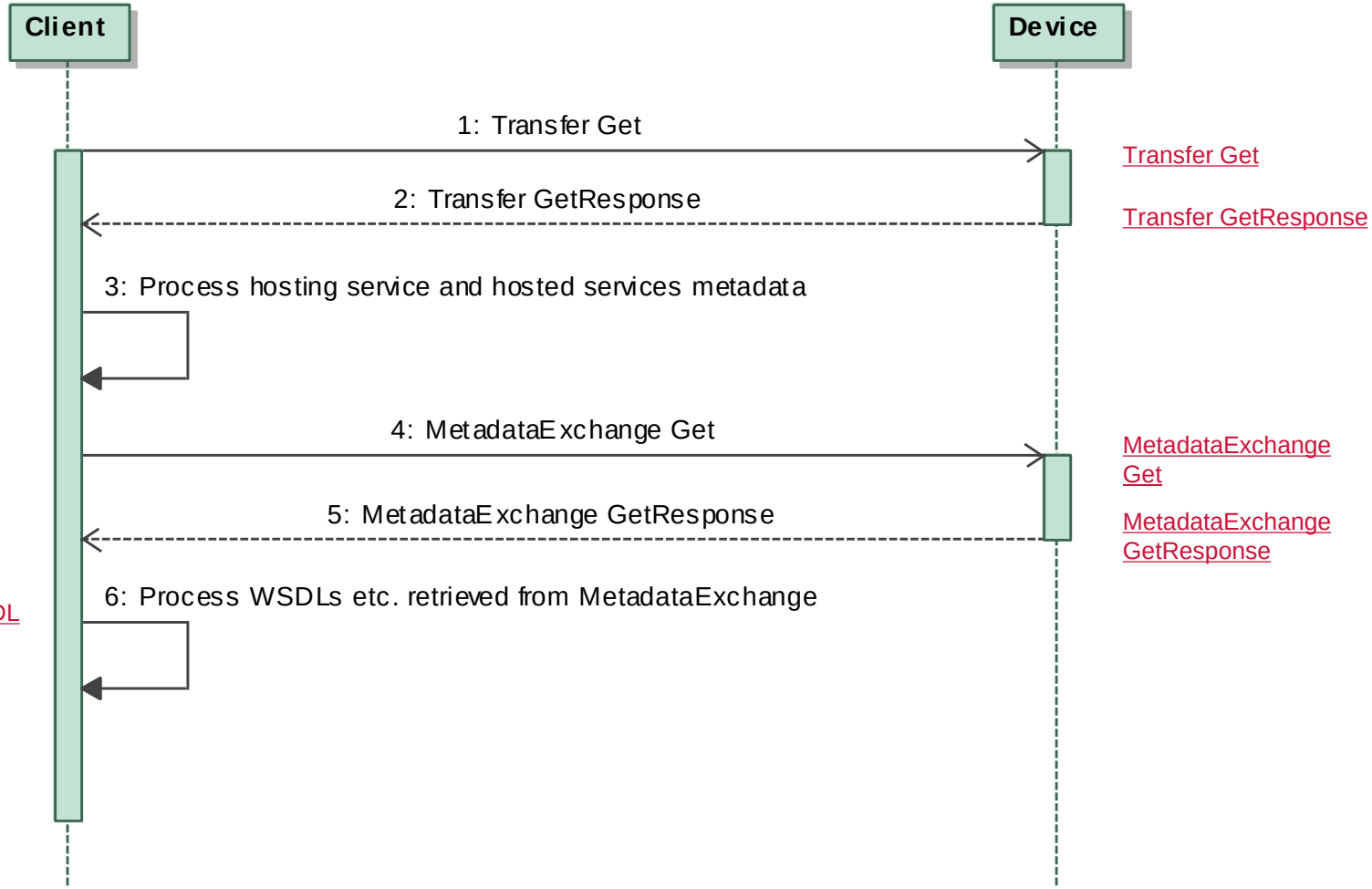
Explicit Discovery



DPWS
Interface Description



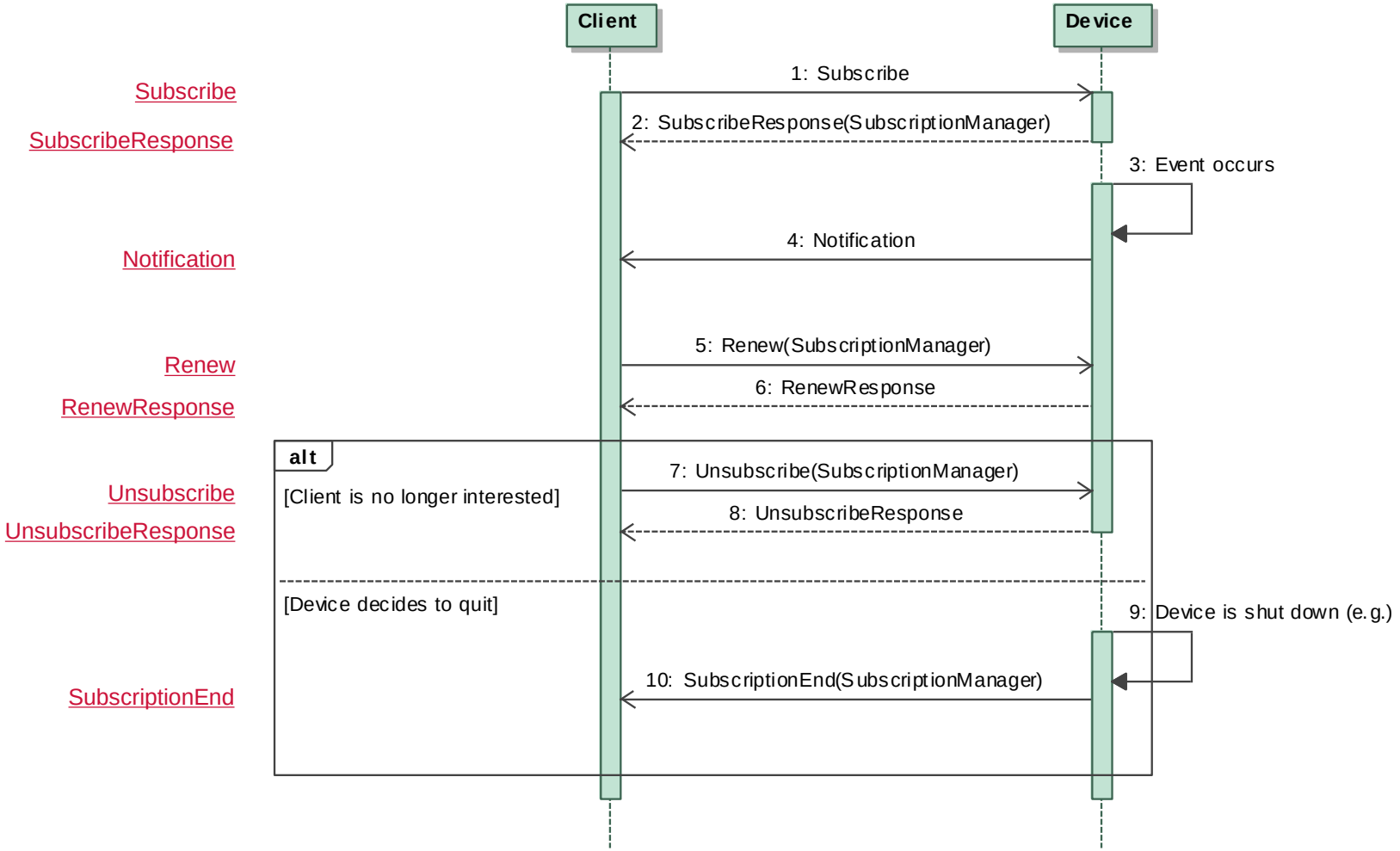
DPWS
Interface Description



DPWS
Event propagation

- In order to support event-driven communication DPWS includes WS-Eventing
- WS-Eventing describes a Web Service-based publish-subscribe pattern
- A Subscription Manager acts as session between an Event Source (aka Hosted Service) and an Event Sink (aka Client)
- After a session is negotiated, the Event Source sends Notifications to the Event Sink

DPWS
Event propagation



DPWS
Secure information transmission

Securing data exchange between Devices and Clients is twofold:

1. Any TCP-based message exchange can be secured using HTTPS (TLS over SSL)
2. Any UDP-based message exchange is not supposed to be encrypted, but may be secured against integrity attacks by using Compact Signatures

→ Authentication only; there is no predefined mechanism to support authorization

MDPWS

MDPWS
Overview

- Numeric data streaming
  - Maybe unreliable
  - Multiple receivers

Waveform Streaming

- Safe & reliable data exchange
- Watchdogs

Patient Safety

- Efficient XML Interchange, if needed

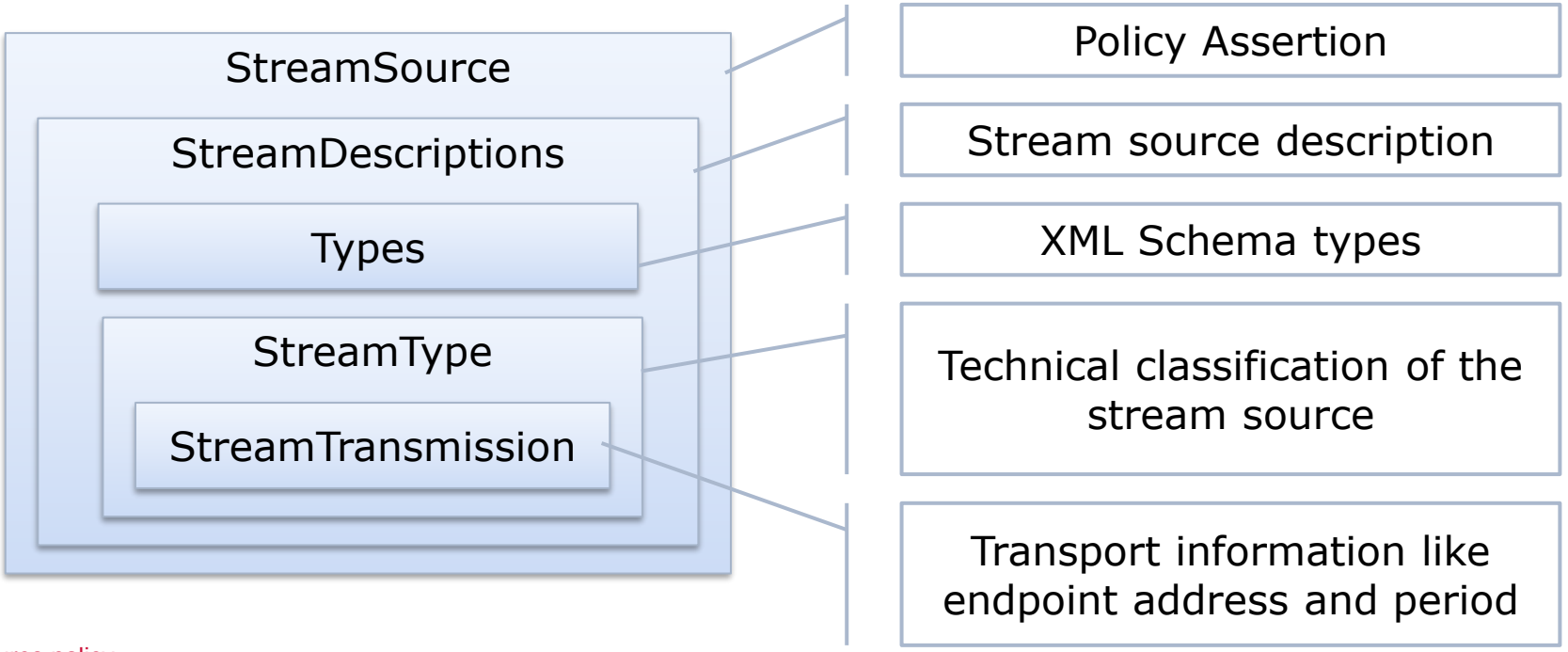
Compression

- Policy assertions
- Limit options in DPWS spec
- New Device Type for Discovery

General

MDPWS
Streaming

- MDPWS defines a WS-Policy assertion in order to indicate streaming support
- The policy can be embedded into WS-MetadataExchange GetResponse messages



StreamSource policy

MDPWS
Patient Safety – Dual Channel Transmission

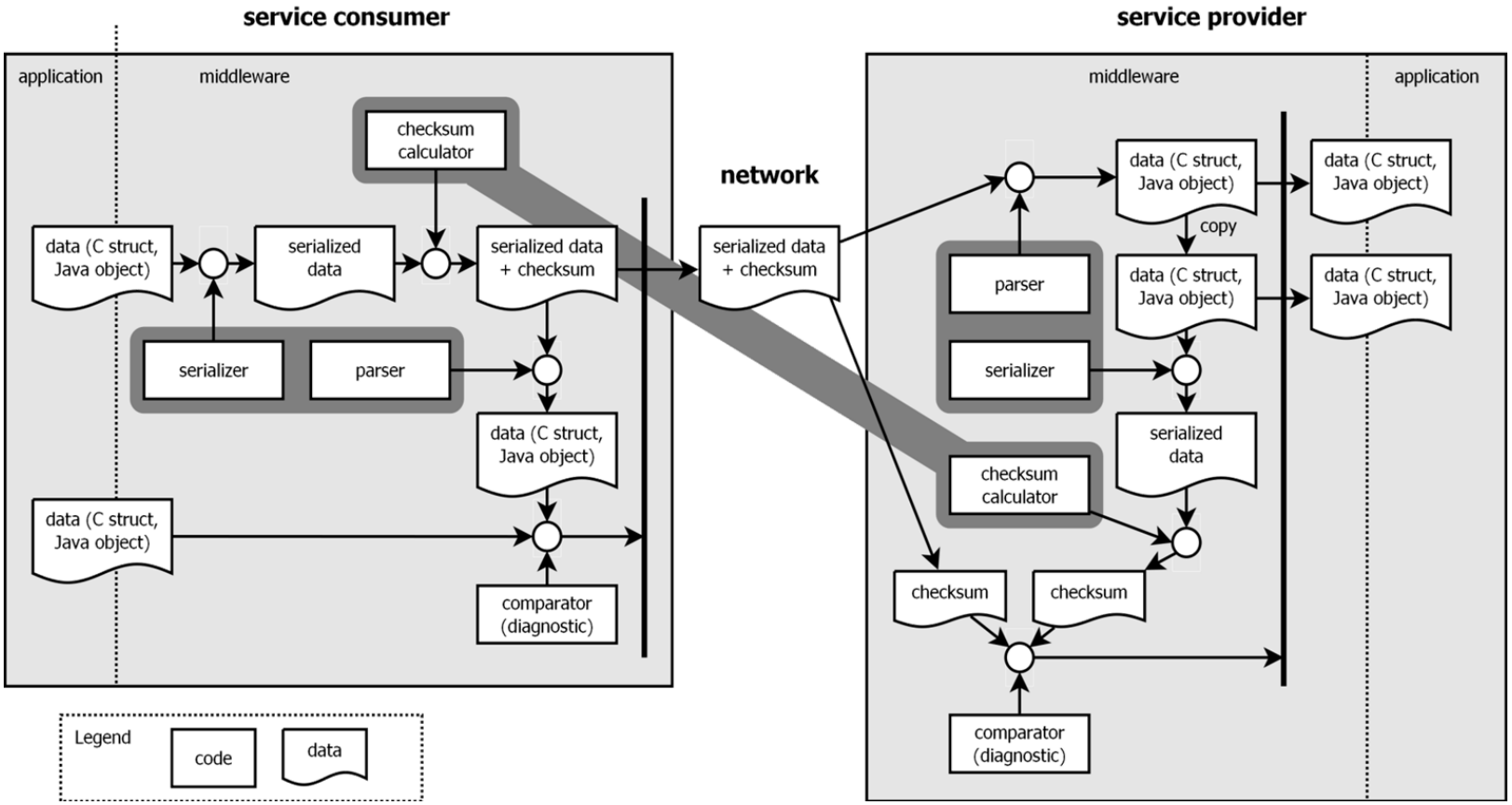
- According to IEC 60601-1, a communication middleware for remote control should ensure single fault safety

Single Fault Safety
characteristic of medical equipment or its parts whereby it remains free of unacceptable risk during its expected service life under single fault condition

Single Fault Condition
condition in which a single means for reducing a risk is defective or a single abnormal condition is present utilization of a dual channel architecture

MDPWS
Patient Safety – Dual Channel Transmission

- The service provider detects a failure, e.g., by means of an invalid checksum.



MDPWS
Patient Safety – Dual Channel Transmission

- How does it look on the wire?

```
<s12:Envelope ...><s12:Header><wsa:Action>...</wsa:Action><wsa:MessageID>...</wsa:MessageID><wsa:To>...</wsa:To>
    <mdpws:SafetyInfo>
      <mdpws:DualChannel>
        <mdpws:DcValue ReferencedSelector="SELECTOR_1">
          7d836f4befca2bda3e8abb1f7bd93345a5b10ae9
        </mdpws:DcValue>
        <mdpws:DcValue ReferencedSelector="SELECTOR_2">
          8dce170de238b1feda2ecd9674ea3ca0d068fbcb
        </mdpws:DcValue>
      </mdpws:DualChannel>
    </mdpws:SafetyInfo>
  </s12:Header><s12:Body>
    <msg:SetString>
      <msg:OperationHandleRef>
        op1
      </msg:OperationHandleRef>
      <msg:RequestedStringValue>
        Value
      </msg:RequestedStringValue>
    </msg:SetString>
  </s12:Body></s12:Envelope>
```

MDPWS
Patient Safety – Safety Context

- Used in remote control to add contextual information to the operation being executed
- Example
  - Client Alice wants to change a parameter on device Bob
  - Bob has his current state of the MDIB that can differ from the latest MDIB Alice is synchronized with (→ IP is best effort)
  - When Alice makes an operation call Bob can impose Alice to attach data to the request – like the current associated patient context – in order to verify if Bob and her are connected to the same patient
  - If that data does not match Bob‘s latest MDIB state, he might refuse the request

MDPWS
Patient Safety – Safety Context

- How does it look on the wire?

```
<s12:Envelope ...><s12:Header><wsa:Action>...</wsa:Action><wsa:MessageID>...</wsa:MessageID><wsa:To>...</wsa:To>
    <mdpws:SafetyInfo>
      <mdpws:SafetyContext>
        <mdpws:CtxtValue ReferencedSelector="SELECTOR_3">
          262656
        </mdpws:CtxtValue>
        <mdpws:CtxtValue ReferencedSelector="SELECTOR_4">
          Sample safety context value
        </mdpws:CtxtValue>
      </mdpws:SafetyContext>
    </mdpws:SafetyInfo>
  </s12:Header><s12:Body>
    <msg:SetString>
      <msg:OperationHandleRef>
        op1
      </msg:OperationHandleRef>
      <msg:RequestedStringValue>
        Value
      </msg:RequestedStringValue>
    </msg:SetString>
  </s12:Body></s12:Envelope>
```

Summary

- MDPWS is a Web Service-based solution to facilitate syntactical interoperability for medical device connectivity
- Clients can detect Devices by using WS-Discovery
- DPWS enables publish-subscribe via WS-Eventing, which provides a connection token between an Event Source and Event Sink
- MDPWS enhances DPWS mainly by
  - streaming capabilites
  - safety information
    - dual channel transmission
    - safety context provision

Thank you for your attention!

Contact information

David Gregorczyk

david.gregorczyk@draeger.com