



MATERNA
WS4D-JMEDS
Addons

Jannis Müthing
jannis.muething@materna.de
MATERNA GmbH

Contents

- I. Introduction
- II. Efficient XML Interchange (EXI)
- III. Communication Managers:
 - I. UPnP and
 - II. Bluetooth
- IV. Presentation URL
- V. Authorization
- VI. DPWS Subnetproxy

Introduction

- JMEDS modularity enables:
 - Staying lightweight
 - Going beyond DPWS
 - Extensibility

- This presentation:
 - from improvements of the current (EXI)
 - over additions (Communication Managers)
 - to solutions for more specific challenges

Efficient XML Interchange (EXI)

EXI

- Compression of XML encoded content
- Binary representation; similar to a Huffmann encoding
- Every DPWS message consists of XML content
- Reduction in size leads to
 - Less congestion of the network
 - Better performance esp. for low end devices

Efficient XML Interchange (EXI)

- Which parts does JMEDS implement?
 - Schema less
 - Byte-Alignment (EXI compression)
 - Schema informed
 - Bit-Alignment
 } Performance improvements
- “EXI Compression” has not been implemented (yet)

- Results:
 - Great!
 - Byte-Aligned/Schema less EXI better than GZIP
 - Bit-Aligned/Schema informed EXI performed best
- Problematic: Schema informed EXI + dynamically generated schemas

Efficient XML Interchange (EXI)

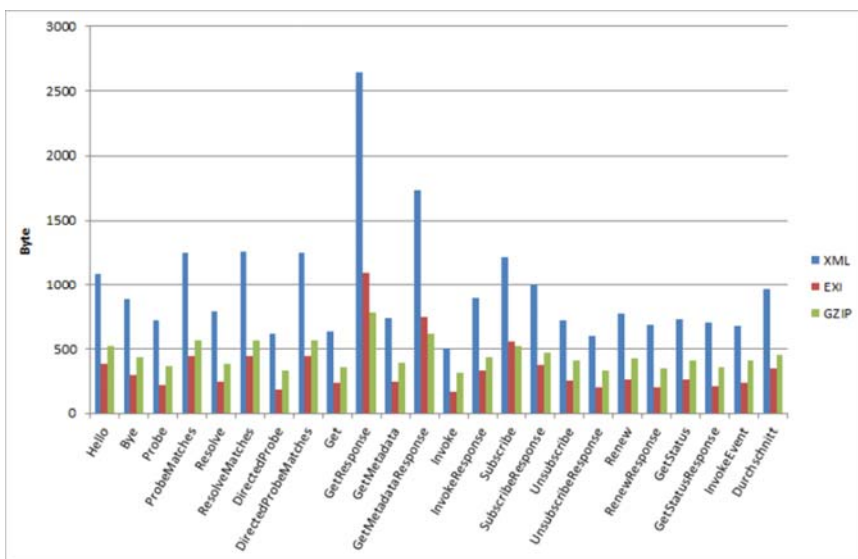


Fig.: Comparison between EXI, GZIP and untouched XML

UPNP

- Differences to DPWS and highlights
 - Not everything is XML
 - Discovery relies on HTTP over UDP
 - Eventing based on variables and per service
 - Multicast eventing

- How to teach a DPWS stack to speak UPNP
 - Receiving UPNP messages
 - Translating into an internal representation
- Target: hiding underlying technology as best as possible
- Users of the API see devices/services not technologies

	DPWS	UPNP
Discovery	Probe	M_SEARCH
	ProbeMatches	HTTP OK
	Bye/Hello	Notify
Metadata exchange	Get/GetMetadata	Accessing resources via HTTP

Fig.: Examples of messages that serve similar needs

Bluetooth

- Similar tasks as the UPNP communication manager
- Hiding the technology etc.
- No eventing at all

- Using BlueCove (open source/multiplatform <http://bluecove.org/>)

	DPWS	Bluetooth
Discovery	Probe	Inquiry (answered only by configured devices)
	ProbeMatches	Inquiry-Response
	Bye/Hello	Inquiry (passive) An Inquiry from a device tells other devices about that device
	Resolve	Inquiry when answered generating internal ResolveMatches
Metadata exchange	Get	LMP_name_req SDP_SearchServiceAttributeReq
	GetMetadata	No similar message BT is based on fix profiles

Fig.: Examples of messages that serve similar needs

Presentation URL

- Web interface for DPWS devices/services
 - Modern web technology:
 - Client/Browser side styling (XSLT/CSS/JS)
 - Device supplies dynamic XML, static XSLT

- Displays metadata e.g.: name, vendor etc.
- Displaying input and output of operations (even complex types)
- Eventing support utilizing AJAX
- Support for attachments (receiving/sending)
 - No real support for streaming
- WSDL inspection

Presentation URL

The screenshot shows a web service interface for 'Zegarek SE'. At the top, there's a header with 'WS4d' and 'Zegarek SE' on the left, and 'MATERNA' on the right. Below the header, there's a table with service details:

Name	Value
Service Type(s)	{http://www.ws4d.org}AdminService
EndpointReferenceAddress(es)	http://139.2.61.217:6652/adminlockservice
Metadata Endpoint Reference Address(es)	http://139.2.61.217:6652/adminlockservice/wsdl.xml
Service ID	http://www.ws4d.org/adminservice

Below the table, there's a list of operations/events:

- DeviceStop
- StopRemoteDebug
- ServiceStop (ServiceList)
- SendInfo
- ShutdownStack
- ServicePause (ServiceList)
- Uptime GetUptime
- SendBye
- MemInfo GetMemInfo
- ServiceStart (ServiceList)
- RemoteStream RemoteDebug (Split)
- DeviceProperty GetProperties
- DeviceRestart

The 'DeviceProperty GetProperties' operation is expanded, showing the following output:

```

Output
  DeviceProperty | DevicePropertiesType
  * attribute1
  * attribute2
  * property
  * attribute1
  * attribute2
  * Name
  * Value
  * Property
  * Property
  
```

Each output item has a corresponding data type and description:

- attribute1: anySimpleType, Container for Property 1
- attribute2: anySimpleType, Part of schema defined
- property: anySimpleType, Container for Name and
- attribute1: anySimpleType, Part of schema defined
- attribute2: anySimpleType, Part of schema defined
- Name: string, Name1
- Value: string, Value1
- Property: string, Property 1
- Property: string, Property 2

Fig.: Screenshot:
Presentation URL

Authorization

Authorization

- Authorization addon support enables fine grained security
- Implementations are asked when receiving messages
- User credentials (username/password) are sent with the messages
- Raising an AuthorizationException when credentials are
 - not recognized or
 - operation was not permitted for that user

Authorization: Default implementation

- The default authorization manager implementation
 - Separating only between services and devices
- Operations are authorized with the service's credentials
 - Setup via XML file ("encryption.xml")
 - Device EPR and Service Id for identification

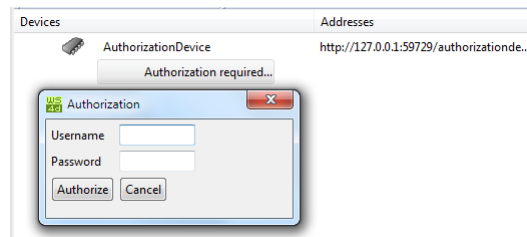


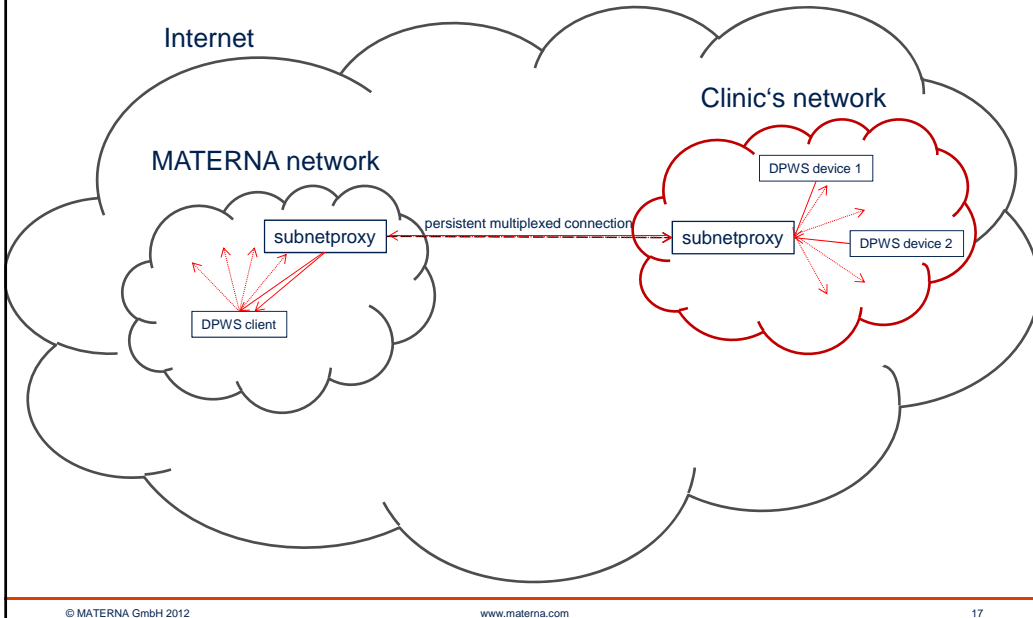
Fig.: Screenshot: JMEDS Explorer 3's new authentication dialog

DPWS Subnetproxy

Subnetproxy

- Connecting remote private networks via a common internet
- Enabling DPWS communication through a single tunnel
- Designed to work in real world restricted network environments:
 - Allowing only incoming connections (firewalls...)
 - Networks might use proxy server for all outgoing connections
- **It's not a discovery proxy!**

DPWS Subnetproxy example: discovery process



DPWS Subnetproxy

- Discovery messages are forwarded/broadcasted in both subnets
- On the destination subnetproxy's side:
 - Exchanging relevant device/service addresses
 - with addresses registered with that proxy's HTTP server
- Incoming connection requests are handled by so called Proxy Entries
 - Proxy Entry ↔ original url
- HTTP connections are made through a multiplexer
- Get/GetMetadata-Response messages are relocalized
- Invocations or simple HTTP GET requests stay untouched