Browser based OPC UA

Uwe Steinkrauss (ascolab GmbH)
Content

- Motivation
- Solution
  - Native, WebServer, JavaScript
- JSUA
  - Protocol Selection, Building Blocks, Architecture
- Validation
  - Timing, Speed, different Browsers
- Conclusions and Future Perspective
**Motivation**

- Mobile devices gain more importance in industrial applications
- New technologies make browser based applications more attractive

```
new technologies
```

```
Mobile Devices act as UA Client
```

```
OPC UA Servers
```

source: www.ascolab.com
Solution - native

- Native implementation
  - Requires client-side installation (App-Store)
  - New implementation for each mobile target (iPhone, Android, WinRT, etc.)
  - Use of declarative GUI languages (e.g. QML)

Pro: highly efficient, direct use of UA-TCP/Binary
Con: 3 times implementation effort

Source: www.ascolab.com
Solution - WebServer

- Proxy/Gateway
  - Webserver “hides” the OPC UA Server
  - Webserver generates (dynamic) websites and fills with OPC UA content (Webserver is “native” UA Client)
  - Protocol conversion for e.g. WebSocket

Pro: sophist. Webserver, capsule more protocols
Con: need for proxy installation, security, conversion, speed
## WebServer - Central Data Portal

<table>
<thead>
<tr>
<th>Display Device</th>
<th>running all kind of web browsers</th>
</tr>
</thead>
<tbody>
<tr>
<td>• PC</td>
<td>![Browser Icons]</td>
</tr>
<tr>
<td>• Mobile Phone</td>
<td></td>
</tr>
<tr>
<td>• Tablet</td>
<td></td>
</tr>
</tbody>
</table>

**Web Clients**
- Browser: HTML, SVG and JAVA Script
- Data: XML, JSON or Binary
- Transport: http or https

### OPC UA Server

- PLCs
- Field Devices
- Access Control

**Web Server**
- Server side scripting: PHP, Ruby, Perl, Python

**OPC UA Client**

- Web Server

- [Source](www.ascolab.com)
Solution – Java Script

- Java Script
  - UA Server must “deliver” the Script-Code (same origin)
  - Only HTTP(S) possible (no direct access to TCP)
  - No message security in Browser (speed)

Pro: No installation on client side, no extra (Proxy)-Webserver required, out of the box functionality

Con: Java Script library for OPC UA Binary Encoding (JSUA) is needed
WebClient - OPC UA Client

Access with native OPC UA clients works too.

Data: UA Binary Protocol: https

HTML, Graphics

Web Server

Field Device

native UA Client

JSUA Client

Mobile, Laptop, PC

JAVA Script based OPC UA Client

source: www.ascolab.com
Protocol Selection - JSUA

- Hybrid profile is best choice for implementing JavaScript based OPC UA Client

Serialization Layer:
- Binary Profile
  - Binary Encoding
- Hybrid Profile
  - Binary Encoding
  - HTTPS

Security Layer:
- UA Secure Conversation

Transport Layer:
- UA/TCP
- HTTPS

WebService
- SOAP/HTTP
  - XML Encoding
  - rarely used in embedded devices

need to implement

native or plugin required

handled by Browser

source: www.ascolab.com
Building Blocks - JSUA

- OPC UA Hybrid Protocol (https-uabinary)
  - Binary en-/decoding of messages performed in pure JavaScript
  - Security mechanisms (HTTPS) applied by the browser
- Delivery of Webpage including Script Code (same origin policy)
Architecture - JSUA

- **Framework Layer**
  - High-level API

- **Serialization Layer**
  - JavaScript object definitions for each of the requests and responses
  - object offers an interface to invoke binary en-/decoding

Source: www.ascolab.com
Architecture - JSUA

- **Transport Layer**
  - Sending asynchronous HTTP(S) requests
  - Receiving response messages

- **Built-in support for long polling (subscriptions)**
Validation

- Performance (Read Service)
  - Comparison: Native (C++) vs. Browser-based (local setup)

![Graph showing performance comparison between Native (C++) and Browser-based (local setup) for different number of nodes. The graph indicates that the time in ms increases linearly with the number of nodes. Source: www.ascolab.com]
Validation

- Performance (Read Service)
  - Comparison: Different devices (remote setup)

![Graph showing Performance (Read Service) comparison between different devices](source: www.ascolab.com)
Validation

- Browser compatibility
  - Works with the current versions of all major browsers (desktop and mobile) – including IE10+

- Support of UA-Subscriptions

- Generic OPC UA-Browser built on top of JSUA
  - Browsing address spaces
  - Reading/Subscribing variables
  - Supports commissioning and maintenance tasks
Demonstration

Source: www.ascolab.com
Conclusion

- JSUA enables direct browser-based access to OPC UA servers
  - No plug-ins needed
  - No proxy servers needed…
  - …but supported for use with larger infrastructures/sophisticated web servers
- JavaScript fast enough to handle binary en-/decoding (response time < 100ms)
Future Perspective

- Moore’s Law
  - Increased CPU speed in mobile devices
  - Acceleration of script engines in browsers
- Server-side JavaScript (e.g. Node.JS)
  - JSUA for OPC UA servers and proxies
Thanks for Your Attention!

Uwe Steinkrauss
Executive Director

ascolab GmbH
Am Weichselgarten 7
D-91058 Erlangen
Phone +49-9131-691-120
info@ascolab.com