

SDPng Update

draft-ietf-mmusic-sdpng-06.txt

Dirk Kutscher	dku@tzi.org
Jörg Ott	jo@tzi.org
Carsten Bormann	cabo@tzi.org

Overview

- **Addressed open issues of –06**
- **Examples**
- **Next steps**

SDPng Structure

Potential Configurations	List of capabilities as XML elements. Only these are processed by capability negotiation.
Definitions	Define commonly used parameters for later referencing.
Actual Configurations	Actual configurations as alternatives for each component.
Constraints	Reference configurations and express constraints on combinations
Session-Level Info	Elements for meta information on individual applications (i.e., streams, sessions), referencing configuration definitions.

draft-ietf-mmusic-sdpng-06.txt

- **Capability description model**
 - Fixed set of capability types
- **Capability negotiation model**
 - Employing RFC 2533 algorithm
- **Worked on some open issues**
 - Concrete XML syntax
 - Simplified model for specifying configurations
 - Specifying transport parameters
 - Formal definition mechanism
 - Package concept
 - Library concept

Capability Model

- **Three different types**
 - **Tokens:**
 - `encoding=PCMU`
 - **Ascertain identity || fail**
 - **Token lists:**
 - `sampling-rate=8000,16000, 44000`
 - **Determine common subset || fail**
 - **Numerical Ranges**
 - `6 <= bitrate <= 64`
 - **Determine common sub-range || fail**
- **Distinguish *optional* capabilities**
 - `silence-suppression supported`
 - **Applicable to each type, failing results in removing the capability, interoperability still possible**

XML Syntax (1)

- **Feature independent negotiation**
 - **Process capability descriptions without knowing semantics**
 - **Access to schema definition not required**
- **Different solutions considered:**
 - **Old compact syntax (-06) had some issues**
 - `<audio:codec name="pcmu" encoding="PCMU" channels="[1,2]" sampling="[8000,16000]"/>`
 - **Requires parsing beyond XML**
 - **Cannot leverage XML mechanisms for formal schema definitions**

XML Syntax (2)

- **Capabilities**

- A collection of independent definitions
- Each definition is processed independently
- Every property is a single XML element
 - Tokens and token lists as element content
 - Numerical ranges with explicit XML attributes
 - No further substructure
 - Descriptions are still standalone

```
<audio:codec name="avp:pcmu">  
  <audio:encoding>PCMU</audio:encoding>  
  <audio:channels>1 2</audio:channels>  
  <audio:sampling>8000 16000</audio:sampling>  
  <audio:bitrate min="6" max="64"/>  
  <audio:silence-suppression status="opt"/>  
</audio:codec>
```

Formal Schema Definition

- **Base specification**
 - SDPng XML document structure
 - Basic data types (token, token lists, ranges)
 - XML-Schema as a definition mechanism
- **Package definitions**
 - Application specific vocabulary
 - Each package definition in unique XML namespace
 - XML-Schema as a definition mechanism

Sample Package Definition

```
<xsd:complexType name="audio:CodecT">
  <xsd:complexContent>
    <xsd:extension base="sdpng:Definition">
      <xsd:sequence>
        <xsd:element name="encoding" type="sdpng:token"/>
        <xsd:element minOccurs="0" name="channels"
          type="sdpng:tokenlist"/>
        <xsd:element minOccurs="0" name="sampling"
          type="sdpng:tokenlist"/>
        <xsd:element minOccurs="0" name="bitrate"
          type="sdpng:range"/>
        <xsd:element minOccurs="0" name="silenceSuppression"
          type="sdpng:optToken"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="audio:codec" type="audio:CodecT"
  substitutionGroup="sdpng:definition"/>
```

Specifying Configurations (1)

```
<cap>
```

```
  <audio:codec name="avp:pcmu">
```

```
    <audio:encoding>PCMU</audio:encoding>
```

```
    <audio:channels>1 2</audio:channels>
```

```
    <audio:sampling>8000 16000</audio:sampling>
```

```
    <audio:bitrate min="6" max="64"/>
```

```
    <audio:silence-suppression status="opt"/>
```

```
  </audio:codec>
```

```
  <rtp:udp name="rtpudpip6">
```

```
    <rtp:network>IP6</rtp:network>
```

```
  </rtp:udp>
```

```
</cap>
```

Specifying Configurations (2)

```
<cap>  
  <audio:codec name="avp:pcmu"> [...]</audio:codec>  
  <rtp:udp name="rtpudpip6"> [...] </rtp:udp>  
</cap>
```

```
<def>  
  <rtp:udp name="rtp-cfg1" ref="rtp:rtpudpip6">  
    <rtp:ip-addr>::1</rtp:ip-addr>  
    <rtp:port>9456</rtp:port>  
    <rtp:pt>1</rtp:pt>  
  </rtp:udp>  
</def>
```

Specifying Configurations (3)

```
<cap>
  <audio:codec name="avp:pcmu"> [...] </audio:codec>
  <rtp:udp name="rtpudpip6"> [...] </rtp:udp>
</cap>
<def>
  <rtp:udp name="rtp-cfg1">[...]</rtp:udp>
</def>

<cfg>
  <component name="interactive-audio" media="audio">
    <alt name="alt1">
      <audio:codec ref="avp:pcmu"/>
      <rtp:udp ref="rtp-cfg1"/>
    </alt>
  </component>
</cfg>
```

Specifying Configurations (4)

- Each *component* (application session) element provides list of alternatives
- Each *alternative* provides definitions for the *component*
 - Referencing definitions from the capability section
 - Providing additional parameters, where required
 - Alternatives that reference non-interoperable definitions are discarded
 - List of definitions
 - No nesting of elements from different packages
 - Semantics are application-specific
 - Applications **MUST** know how to interpret definitions
 - No restrictions on quantity or order

Libraries

- **Libraries:**
 - Pre-defined definitions, e.g., a set of audio codec definitions
 - Referenced from a description document
 - **Semantics difficult to get right**
 - Application-independent negotiation would require access to library definitions
 - Requirement to *include* library definitions into description document
 - Capability negotiation has to consider *all* definitions
- Forego libraries, include definitions inline**

Next Steps

- **draft-ietf-mmusic-sdpng-07**
 - Update document structure spec.
 - Include XML-Schema definitions
- **Applications & Tools**
 - Define packages
 - Publish XML-Schema for base spec and packages