SDPng Update

draft-ietf-mmusic-sdpng-06.txt

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

Overview

- **Changes in –06**
  - **New document structure**
  - **New requirements for capability specifications**
  - **New capability negotiation rules**
  - **Removed XML-Schema-based formal definition**
  - **Removed text on session-level information**
- **Open issues / current work**
- **Next steps**

Changes: Document Structure

1. Introduction
2. Terminology and System Model
3. Capability Negotiation Overview
   - Conceptual Overview, some requirements
4. SDPng Concepts
   - SDPng structure
   - Semantics of potential and actual configurations
   - Syntax mechanisms, requirements for referencing
5. Syntax Definitions
   - How to specify potential configurations
6. Specification of the Capability Negotiation
   - Rules for negotiating capabilities

Changes: Capability Specifications

- **Requirements:**
  - **MUST support feature-independent negotiation**
  - **MUST NOT require access to a package definition in order to process capability descriptions**
- **Solution:**
  - **Fixed set of basic types**
    - **Symbols, symbol sets, numerical ranges**
    - **Optional parameters**
  - **Encode type of capability into value**
- **Example**
  - `<audio:codec name="pcmu" encoding="PCMU" channels="[1,2]" sampling="[8000,16000]"/>`

Changes: Naming of Definitions

- **Requirement:**
  - **MUST specify fully-qualified names for definitions in order to avoid name collisions**
- **Solution:**
  - **Prefix names with namespace-prefixes that map to unique namespace identifiers (analogous to XML namespaces for XML GIs)**
  - **SDPng processors MUST expand prefixes to complete namespace names**
- **Example:**
  - `<video:codec name="dku:codecX"/>`
  - **Where** `dku` **maps to** `http://www.example.com/dku/`

Changes: Capability Negotiation Rules

- Describe general behaviour
- Concrete processing behaviour is implementation-specific
- Specify two alternative procedures
  - Offer/Answer based scheme
    - No “collapsing”, just matching and selection of configurations
  - RFC 2533 (conneg)
    - Complete feature-independent negotiation as per RFC 2533

Capability Negotiation with RFC 2533

- **Translating SDPng into RFC 2533**
  - **Translate each attribute into feature predicate, considering data types**
  - **Aggregate resulting predicates into feature set (conjunction)**
  - **Aggregate multiple feature sets (alternative potential configurations) into disjunction**
- **Negotiation process**
  - **Creating conjunction of feature sets**
  - **Applying RFC 2533 canonicalization**
- **Integrating results into SDPng document**
  - **Match each SDPng element (“capability“) against negotiation result**
  - **Adopt compatible elements only**

Open Issues

- Concrete syntax definition
  - Formal definition mechanism
    - SDPng base spec, packages
- Session-Level-Information
  - Minimal feature set in base spec
- Library concept
  - Always include all definitions?

Open Issue: Syntax Definition

- **SDPng base specification**
  - **General model: potential capabilities, actual capabilities, constraints, meta-info** ✓
  - **Describing and processing capabilities** ✓
  - **Describing actual configurations**
    - **Referring to capabilities**
    - **Adding transport parameters**
  - **Formal syntax definition mechanism**
    - **Update XML-Schema definition**
- **Packages**
  - **Definition mechanism**
  - **Package definitions need to specify application specific element types and attributes**
  - **Distinguish parameters and capabilities**

SDPng Structure

Potential Configurations	List of capabilities as XML elements. Only these are are processed by capability negotiation.
Transport Parameters + Actual Configurations	Actual configurations as alternative for each component. Transport parameters may be factored-out and referenced in configurations.
Constraints	Reference configurations and express constraints on combinations, number of instantiations etc.
Session-Level Info	Elements for meta information on individual applications (i.e., streams, sessions), referencing configuration definitions.

Potential Configurations

```
<video:codec name="dku:codecX"/>
<rtp:udp name="rtpudpip6"
         ipversion="6"/>
```

Alternatives for Components

```
<component name="video1">
  <alt name="codecX">
      <video:codec ref="dku:codecX">
      <rtp:udp ref="rtpudpip6"
       ip-addr="::1" rtp-port="4567"
       rtcp-port="9876" pt="97"/>
</component>
```

Potential Configurations

```
<video:codec name="dku:codecX"/>
<rtp:udp name="rtpudpip6"
         ipversion="6"/>
```

Transport Parameters

```
<rtp:udp name="rtp1" ref="rtpudpip6"
 rtp-addr="::1" rtp-port="4567"
 rtcp-port="9876"/>
```

Alternatives for Components

```
<component name="video1">
  <alt name="codecX">
      <video:codec ref="dku:codecX">
      <rtp:udp ref="rtp1" pt="97"/>
  <alt name="codecY">
      <video:codec ref="dku:codecY">
      <rtp:udp ref="rtp1" pt="98"/>
</component>
```

Next Steps

- Next revision
  - Details on describing actual configurations, transport parameters
    - Will post proposal to mailing list
  - Formal definition mechanism for packages
- Other work items
  - Specific package definitions
  - Session-level information