Session Announcements
(SAP, RFC 2974)

Session Description
(SDP, RFC 2327)
(SDP, RFC 4566)

Slide contributions by Dirk Kutscher (Uni Bremen TZI)

Conference Establishment & Control

Workshop 1. Create
Descr.: IETF-Tag Internet-Multimedia
Orig.: J.Ott jo@tzi.org 327689113
Info: http://www.tzi.org/dmn/
Start: 29.09.2004 / 12:00
End: 29.09.2004 / 12:40
Media: Audio PCM 234.5.6.7/39000
Media: Video H.263 234.5.6.8/29000

2a. Announcement
   Announcement Protocol
   Netnews
   WWW

   2b. Invitation
   E-Mail
   Invitation Protocol

   2c. Inquiry
   Streaming Protocol

3. Join

4. Media streams

3. Join
IETF Multimedia (Conferencing) Architecture

Session Announcement Protocol (SAP)

- Announcing multimedia sessions to a broad audience
- Session announcements contain SDP
  - Subject of the session
  - Date(s) and time(s)
  - Media streams and addresses
  - Further information
- SAP Functions
  - New session announcements
  - Modify announcements
  - Delete announcements
  - Support for relays
- Earlier: Coordinate use of multicast address space
Dissemination of SAP Announcements

- **Scope of Announcements**
  - Per (administratively defined) multicast address scope
  - Local: 239.255.0.0/16
  - Organization local: 239.192.0.0/14
  - SAP conferences: 224.2.0.0 – 224.2.127.253
  - Other: Global
  - Similar considerations for IPv6
    - Scope identifier built-in into the IPv6 address structure

- **SDP descriptions should use addresses of same scope**
  - To ensure that receivers can also receive the media streams if they can receive the announcements
SAP Features

- Limited announcement bandwidth per scope
  - e.g. 4000 bit/s (defined per scope)

- Calculation algorithm roughly similar to RTCP
  - Measure incoming SAP packets per scope
    - Sizes, number of announcements
  - Calculate size of own announcements
  - Estimate available share of bandwidth
  - Calculate own transmission interval
    - Use dithering (± 1/3 of the interval)
    - Timer reconsideration before transmitting

(NeW) Announcements

- SAP uses UDP/IP: no reliability
- Repeat announcements in "regular" intervals
- Intervals: in the order of minutes
  - e.g. minimum 5 min
- Announcements for easy comparison identified by
  - Source IP address (of the creator)
  - 16 bit hash value
- May be authenticated (creator authentication)
- May be encrypted
- May be compressed
- May contain different payload types (SDP is just one)
### SAP Packet Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>Version — =1 for SAPv2</td>
</tr>
<tr>
<td>A</td>
<td>Address type — =0 IPv4 source address — =1 IPv6 source address</td>
</tr>
<tr>
<td>T</td>
<td>Type — =0 Announcement packet — =1 Deletion packet</td>
</tr>
<tr>
<td>E</td>
<td>Encrypted — indicates encryption of the announcement packet</td>
</tr>
<tr>
<td>C</td>
<td>Compressed — indicates that the announcement packet is compressed</td>
</tr>
<tr>
<td>Auth Length</td>
<td>Length of the authentication header (0 = no authentication)</td>
</tr>
<tr>
<td>Originating Source</td>
<td>(32 or 128 bits)</td>
</tr>
<tr>
<td>Optional Authentication Data</td>
<td>...</td>
</tr>
<tr>
<td>Optional Payload (MIME) Type</td>
<td>... — NUL</td>
</tr>
<tr>
<td>Session Description Payload</td>
<td></td>
</tr>
</tbody>
</table>

**Diagram:**

```
0 2 3 7 8 15 16 31
V=1 A O T E C Auth Length Message ID Hash
```

© 2007 Jörg Ott
SAP Header Fields (2)

- Message ID Hash: Unique value per session creator
- Originating Source: IP address of session creator
- Authentication Data: Source Authentication information (PGP and CMS formats defined so far)
- Payload MIME Type: NUL-terminated text string indicating the MIME type of the payload
  Default: application/sdp

Deleting Announcements

- Explicit Timeout
  - No need to announce sessions after the “end time” in SDP
  - Caveat: the SAP receivers and relays need to understand SDP

- Implicit Timeout
  - Receiver observe repetition of announcement
  - After 10 times the announcement interval (or one hours) with re-announcement the session is removed

- Explicit Deletion
  - Send Deletion packet for a session
  - Message ID Hash and Originating Source must match
  - SHOULD be authenticated (match the original announcement)
Modifying Announcements

- Replace an existing session description
  - E.g. modify media or start / end times
  - Update description
- Message ID Hash MUST change
- Modifying announcement MUST be authenticated if and only if the original announcement was
- If in doubt, a new session is “created”
  - Prevent denial-of-service attacks
- If proper match is found, the old session information is simply replaced by the new one

SAP Security

- Encrypted messages for secure information distribution
  - Should be only used with limited size receiver groups
  - Avoid waste of computation resource if many receivers cannot decrypt the message
  - Key distribution out-of-scope
  - Limited applicability, limited usefulness
- Authentication
  - SHOULD always be done
  - Enables at least to verify that two messages are from the same source
  - Proper source authentication requires PKI
- General observation
  - Both is rarely used in practice
  - Current use of SAP in the Internet does not justify the effort...
Session Announcement Tool: SDR

<table>
<thead>
<tr>
<th>Name</th>
<th>Contact</th>
<th>Port</th>
<th>Hong</th>
<th>Gulp</th>
</tr>
</thead>
<tbody>
<tr>
<td>UO Broadcast NASA Videos (2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UO Broadcast NASA Videos (2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UO Broadcast Oregon Story</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UO EDGE News Clips</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UO Medical Management of Biological Casualties</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UO Market</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UO Network Resources</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UO Present at I30U News Conference</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UO Present at I30U News Conference 11:30 PM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SDR: Session Information

- Encrypted: none (NOENC) 10:00-11:00 AM Classical Radio 2001-01-01 10:00 [MD5]

Grouped into three categories

- 1 x session, m x time, n x media

Session Description Protocol (SDP)

- All you need to know about a session to join
  - who? — convener of the session + contact information
  - what about? — name and informal subject description
  - when? — date and time
  - where? — multicast addresses, port numbers
  - which media? — capability requirements
  - how much? — required bandwidth

- Grouped into three categories
  - 1 x session, m x time, n x media

© 2007 Jörg Ott
Session Level Description

v=0 Version
o= Owner / creator of the session + unique identifier + version
u= URL for further information
e= Contact email address
p= Contact phone number
b= Bitrate information
k= Encryption key information
z= Time zone adjustment
a= Attribute lines (for extensions)
c= Connection (=address) information

Time Description

- Start and end time(s) of a session
  - Plus time zone adjustment
- Regular repetitions
  - Every Tuesday and Thursday, 10 – 12
  - Every day
- Arbitrary repetitions
  - Repeated specification of t= lines

t= Start, end time (NTP seconds, special case: 0, 0)
r= Repetitions (interval, duration, offsets)
Media Description

- Define the media streams comprising a conference
  - Media type (audio, video, text, tones, application, ...)
    - Only audio, video, text, tones are well-defined
  - (multicast) address(es) + port number
  - Maps RTP payload types for media to encoding formats
  - Other media level attributes

\[ m= \text{Media and port specification} \]
\[ c= \text{IP address specification (inherited from session)} \]
\[ a= \text{Attributes for this media stream} \]
\[ \text{rtpmap:, fmtp:, recvonly, portrait | landscape} \]

SDP Example

Length of Time represented by Media in a single Packet

(in SIP: address where originator wants to receive data)

\[ v=0 \]
\[ o=lynnch 3117798688 3117798739 IN IP4 128.223.214.23 \]
\[ s=UO Presents KWAX Classical Radio \]
\[ i=University of Oregon sponsored classical radio station KWAX-FM \]
\[ u=http://darkwing.uoregon.edu/~uocomm/ \]
\[ e=UO Multicasters multicast@lists.uoregon.edu \]
\[ p=Lucy Lynch (University of Oregon) (541) 346-1774 \]
\[ a=tool:sdr v2.4k \]
\[ a=type:test \]
\[ n=audio 30554 RTP/AVP 0 \]
\[ \text{c=IN IP4 224.2.246.13/127} \]
\[ a=ptime:40 \]
Session Management Attributes

- Signaling the RTCP port (RFC 3605)
  - Motivation: RTP and RTCP port number may not be adjacent
  - `a=rtcp:<port> [<nettype> <addrtype> <addr>]`
  - `a=rtcp:60004 [IN IP4 192.168.11.12]`

- Signaling multicast sources (IMGPv3, SSM)
  - `a=src-filter:incl IN IP4 232.3.4.5 192.168.1.89`
  - `a=src-filter:excl IN IP4 225.3.4.5 192.168.1.89`

- Session bandwidth (independent of lower layers, RFC 3890)
  - `b=TIAS:64000`
  - `a=maxprate:40.0`

- RTCP bandwidth (modify sender/receiver share, RFC 3556)
  - `b=BS:1600`
  - `b=BR:14400`

---

Session Description and Capability Negotiation

From Session Announcement to Session Invitation
Characteristics of SAP Announcements

- **Common view**
  - Every SAP-receiver sees the same description
    - Session meta information & scheduling
    - Media description & transport parameters

- **Identical transport parameters for all participants**
  - IP-Multicast service model:
    - Senders send to a multicast group (IP address)
    - Receivers join ("tune into") a multicast group

Session Initiation

- **Distribute conference configuration**
  - Applications
    - Media types, media format parameters
  - Transport Parameters
    - IP addresses, transport protocols, protocol parameters

- **Negotiate Parameters!**
  - Heterogeneous end systems
    - Different hardware and software capabilities
  - User preferences

- **SDP provides syntax mechanisms to express parameters**
  - Procedural model for initiation required
Invitation: Conceptual Model

**INVITATION:**
- List of applications
- List of supported configurations

**RESPONSE:**
- List of applications and configurations that are supported by A and B

**Selected Configuration and A’s Transport Parameters**

**B’s Transport Parameters**

- Match A’s configuration with B’s configuration
- Determine B’s transport parameters

Session Initiation with SDP (1)

**A**

```
v=0
o=slice 2890844526 2890844526 IN IP4 host.anywhere.com
s=phone call
c=IN IP4 host.anywhere.com
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Send PCMU-Audio to host.anywhere.com/49170

**B**

```
v=0
o=bob 2890844730 2890844730 IN IP4 host.example.com
s=phone call
c=IN IP4 host.example.com
m=audio 49920 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

OK, send PCMU-Audio to host.example.com/49920
Session Initiation with SDP (2)

SDP Offer/Answer Model (RFC 3264)

- For initiation of unicast sessions
- Objective: generate common view of session configuration
- Simple exchange of capability descriptions

Basic Model:
- A sends offer to B, including
  - Set of media streams and codecs A wishes to use
  - Transport parameters (where A wants to receive data)
- B sends answer to A
  - For each stream in offer, indicating whether stream is accepted or not
  - For each stream add transport parameters (where B wants to receive data)
Codec Selection

- Offer can provide multiple codecs for a media stream.
  - Ordered by preference
  - Offerer commits to support all codecs (one at a time)
  - Answerer should generate list of codecs for each stream, maintaining payload type mapping
  - New codecs may be added

- One of N codec selection
  - Offer multiple codecs, but cannot change dynamically
  - Offerer sends codec list "with reservation"
  - Answerer sends back subset
  - Offerer "locks" one codec for session
  - Implemented with a=inactive media level attribute...
Unidirectional Streams

Offerer

\[v=0\]
\[o=alice 2890844526 2890844526 IN IP4 host.anywhere.com\]
\[s=phone call\]
\[c=IN IP4 host.anywhere.com\]
\[t=0 0\]
\[m=audio 49170 RTP/AVP 0\]
\[a=rtpmap:0 PCMU/8000\]
\[a=recvonly\]

I only want to receive PCMU-Audio. Please send to host.anywhere.com/49170.

Answerer

\[v=0\]
\[o=bob 2890844730 2890844730 IN IP4 host.example.com\]
\[s=\]
\[c=IN IP4 host.example.com\]
\[t=0 0\]
\[m=audio 49920 RTP/AVP 0\]
\[a=rtpmap:0 PCMU/8000\]
\[a=sendonly\]

OK, I will send PCMU audio to you. BTW, for this RTP stream I expect RTCP feedback on port 49920+1 (at host.example.com).

Send/Receive Only

- Media streams may be unidirectional
  - Indicated by \texttt{a=sendonly, a=recvonly}
- Attributes are interpreted from sender’s view
  - \texttt{sendonly}
    - Recipient of SDP description should not send data
    - Connection address indicates where to send RTCP receiver reports
    - Multicast session: recipient sends to specified address
  - \texttt{recvonly}
    - Sender lists supported codecs
    - Receiver chooses the subset he intends to use
    - Multicast session: recipient listens on specified address
- \texttt{inactive}
  - To pause a media stream (rather than deleting it)
Codec Selection

Offerer

```plaintext
v=0
o=alice 2890844526 2890844526 IN IP4 host.anywhere.com
s=phone call
c=IN IP4 host.anywhere.com
t=0 0
m=audio 49170 RTP/AVP 0 2 3
a=rtpmap:0 PCMU/8000
a=rtpmap:2 G721/8000
a=rtpmap:3 GSM/8000
```

Answerer

```plaintext
v=0
o=bob 2890844730 2890844730 IN IP4 host.example.com
s=
c=IN IP4 host.example.com
t=0 0
m=audio 49920 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

Example SDP Alignment

```plaintext
v=0
o=jo 7849 2873246 IN IP4 ruin.inf...s=SIP call
t=0 0
c=IN IP4 134.102.218.1
m=audio 52392 RTP/AVP 98 99
a=rtpmap:98 L8/8000
a=rtpmap:99 L16/8000
m=video 59485 RTP/AVP 31
a=rtpmap:31 H261/90000
```

```plaintext
v=0
o=cabo 82347 283498 IN IP4 dmn.inf...
s=SIP call
t=0 0
c=IN IP4 134.102.218.46
m=audio 49823 RTP/AVP 98
a=rtpmap:98 L8/8000
m=video 0 RTP/AVP 31
```

Resulting configuration:

```
jo@ruin 134.102.218.1
```

```
:52392 audio data L8/8000 :49823 (no video)
```

```
cabo@dmn 134.102.218.46
```

Grouping of m= lines in SDP

- **Observation:**
  - Multiple m= lines in SDP have no relationship to each other
    - Independent media streams
    - Usually different media types

- **Problem:**
  - Want to express synchronization relationship
    - Lip synchronization
  - Concept of "flows" that consist of several media streams
    - Streams encoded in several formats
    - May be streamed from different hosts/ports
    - Useful application in some IP telephony scenarios

---

**Example for Lip Synchronization**

```
v=0
o=Laura 289083124 289083124 IN IP4 one.example.com
t=0 0
c=IN IP4 224.2.17.12/127
a=group:LS 1 2
m=audio 30000 RTP/AVP 0
  a=mid:1
m=video 30002 RTP/AVP 31
  a=mid:2
m=audio 30004 RTP/AVP 0
  i=This media stream contains the Spanish translation
  a=mid:3
```

Stream 1 and 2 should be synchronized.
ANAT Grouping

- Alternative Network Address Types (RFC 4091)
  - Allows expressing IPv4 and IPv6 address alternatives

```
v=0
o=bob 280744730 28977631 IN IP4 host.example.com
s=
t=0 0
a=group:ANAT 1 2
m=audio 25000 RTP/AVP 0
c=IN IP6 2001:DB8::1
a=mid:1
m=audio 22334 RTP/AVP 0
c=IN IP4 192.0.2.1
a=mid:2
```

FEC Grouping

- Group basic and FEC data (draft-ietf-mmusic-fec-grouping-05.txt)

```
v=0
o=adam 289083124 289083124 IN IP4 host.example.com
s=ULP FEC Seminar
t=0 0
c=IN IP4 224.2.17.12/127
a=group:FEC 1 2
a=group:FEC 3 4
m=audio 30000 RTP/AVP 0
a=mid:1
m=application 30002 RTP/AVP 100
a=rtpmap:100 ulpfec/8000
a=mid:2
m=video 30004 RTP/AVP 31
a=mid:3
m=application 30004 RTP/AVP 101
c=IN IP4 224.2.17.13/127
a=rtpmap:101 ulpfec/8000
a=mid:4
```
Further Groupings

- Alternative RTP profiles
  - Dealing with combinatorial explosion of options
  - E.g. AVP and AVPF, AVP and SAVP

- Layered coding and scalable (video) coding
  - Convey dependencies across different RTP sessions

- Alternative addresses?
  - Multicast vs. unicast distribution, e.g. for a media streaming offer

- ... 

- Currently design team formed
  - Resulting solution may but need not use grouping

Simple Capability Declaration in SDP

- Observation:
  - Capability negotiation/declaration in SDP too limited
  - Session description describe both session parameters and capabilities without clear distinction
  - Simultaneous capability restrictions cannot be expressed
    - “Supporting multiple codecs for one media type, but only one per session”

- Simcap: add SDP attributes to explicitly express capabilities
Simcap Example

Sender is willing to receive and send G.729 (18) and telephone-events.

Additionally, it declares the following capabilities:
• PCMU-Audio (0)
• telephone-events (different events)
• Fax-Relay over UDP and TCP

Semantics:
• a=sqn: declares a sequence number
• a=cdsc: declare one or more capabilities
• a=cpar: additional parameters for a declaration
Connection-oriented Media with SDP

- Focus on TCP (RFC 4145) and TLS (RFC 4572)
- In contrast to UDP, a connection must be established
  - Who is to initiate setup, who is to listen?
    - `a=setup: active | passive | actpass | holdconn`
  - What if a connection already exists (e.g., when renegotiating)
    - Keep the existing connection?
    - Set up a new one?
    - `a=connection: new | existing`
  - When to tear down a connection?
    - If a “new” one is specified, close an existing one
- Relies on interactive agreement on how to proceed

Labeling media streams

- Unique identification
  - Across SDP session descriptions
    - Contrast to mid (which is valid within a session only)
  - `a=label:<token>`
  - No semantics
- Attaching stream semantics
  - Usually relevant within an SDP session
  - Hint at stream semantics
    - E.g., if multiple media streams are received: which is which?
  - `a=content:<token>`
  - `token=slides | speaker | sl | main | alt | user-floor | ...`
SDP Extensions: There is more…

- Precondition signaling for media streams
  - Security
  - QoS
  - Connectivity

- Key management (fixing $k=\cdot$)
  - End-to-end key negotiation
  - End-to-end key distribution (via a protected channel)

- And support for further media types
  - Multicast file distribution, application sharing, …

- Will be discussed in the context of signaling protocols

Summary So Far

- SDP syntax can be used for session initiation
  - But requires additional specification of procedures: Offer/Answer

- SDP & Offer/Answer not appropriate for all usage scenarios
  - Fundamental SDP problem of combining configuration descriptions with capability declaration
  - Lack of expressiveness: grouping of media streams
  - "$a=\cdot$" only a limited extension mechanism

- SDP Syntax
  - Limited expressiveness and cumbersome extensibility
SDP Syntax Issues

- **Basic** set of description elements for media sessions
  - IP addresses, port numbers, RTP payload types, parameters

- **Extensibility:** new session / media level attributes
  - `a=<keyword>:<value> ...
  - Senders can use arbitrary attributes:
    - Important attributes cannot be distinguished from unimportant ones
    - Name clashes (misinterpretation) cannot be excluded
  - In principle, allows for any kind of extension
    - Grouping, constraints, ...

- SDP workarounds rather clumsy, inefficient, ...

Fixing SDP…

- The grand idea (in 1999): SDPng
  - More expressiveness
    - For individual media and their combination
    - Often only very basic media descriptions available
  - Real negotiation functionality
  - Extensibility
  - More explicit (e.g., semantics for media sessions)

- Major issue: syntax choice (XML)
  - Not backwards-compatible (deployment, vendor know-how, code re-use)
  - Back in the late 1990s, XML considered “too expensive” for endpoints

- Result: no buy-in from vendors → little motivation → dead

- But: conceptual elements survived
SDP Next Generation (SDPng)

- Being designed to address SDP’s flaws...
  - Limited expressiveness
    - For individual media and their combination
    - Often only very basic media descriptions available
  - No real negotiation functionality
  - Limited extensibility (clumsy, hard to coordinate)
  - No semantics for media sessions (only implicit)
  - Also: Avoid second system syndrome!
    - Simple, easy to parse, extensible, limited scope

- Major issue: syntax choice (XML)
  - Not backwards-compatible (deployment, vendor know-how, code re-use)
  - Back in the late 1990s, XML considered “too expensive” for endpoints

- New approaches to capability negotiation presently discussed

Intelligent Endpoints

- Intelligent endpoints with support for
  - Multiple codecs and format parameters
  - Different applications (e.g., audio, DTMF, video, games)
  - Many transport parameters
    - RTP/UDP/IPv4, RTP/UDP/IPv6, Security, Source-Specific-Multicast...
  - AAA & security parameters

Must be expressible in configuration descriptions!
Intelligent Endpoints

- Heterogeneous end systems
  - Different capabilities
  - Different user preferences
  - Dynamic configuration

Interoperability requires dynamic negotiations of parameters!

Specific Requirements

- **Expressiveness**
  - Describe all *required* configuration parameters

- **Extensibility**
  - No fixed parameter set
  - Profiles (“packages”) for new configuration parameters

- **Support for Negotiation**
  - Derive commonly supported configurations from individual configuration descriptions (for \( n \geq 2 \))

- **Compatibility**
  - Drop-in replacement for SDP in SIP applications
SDPng’s Conference Model

- Components in a conference
  - Individual cooperation functions
  - Characterized by the service they provide (not by their technical implementation)

- Implementations of components
  - Depend on endpoint capabilities and user preferences
  - Use of implementations must be configured or negotiated

Potential Configurations

- Configurations for implementing a component
  - Common capabilities
  - Not a complete conference description, e.g., no transport parameters
  - Dynamic set of parameters
    - Can change over the course of a conference
Actual Configurations

- Complete specification of conference parameters
  - Selected subset of potential configurations
  - Complemented with
    - Media format parameters
    - Transport parameters

```
address=192.168.1.1
port=37000
codec-type=PCMU
payload-type=0
```

SDP Capability Negotiation

draft-ietf-mmusic-sdp-capability-negotiation-07.txt

- Four elements
  - Definition of capabilities
  - Proposing potential configurations
  - Agreeing on actual configurations
  - Negotiation process
    - Based upon the SDP offer/answer model

1) Generate
2) Send
3) Process & match
4) Choose & reply
5) Process
Mapping to SDP…

Reminder

\[
\begin{align*}
  m= & \text{audio} 54321 \text{ AVP/RTP 0 8 96} \\
  a= & \text{rtpmap:96 g729} \\
  a= & \ldots \\
  m= & \text{video} 54545 \text{ AVP/RTP 32} \\
  a= & \ldots
\end{align*}
\]

Requirements

- Must be expressed in SDP syntax
- Backwards compatibility
- Operate in one round-trip (offer/answer exchange)
- Extensible
- Not too verbose (messages can already grow quite large)
- ...

Basic Approach and Syntactic Elements

- Backwards compatibility leaves SDP attributes as the only option
- Extensibility: feature tags
  - Supported: \texttt{a=csup:foo,bar,crunch}
  - Required: \texttt{a=creq:zompel}
- Capability descriptions
  - Transport capability: \texttt{a=tcap:<n> RTP/AVP}
  - Media level attribute: \texttt{a=acap:<m> rtpmap …}
- Configuration negotiation
  - Potential configuration: \texttt{a=pcfg:<k> <n> <m>}
  - Actual configuration: \texttt{a=acfg:<k> <n> <m>}
- Offer/answer extension allowing to include capabilities
Litmus Test Example: Optional Security

- Offerer supports secure media streams (preferred)
  - Yet, wants to allow fallback to insecure communications for compatibility
  - Does not want to wait for an extra round-trip
More Syntax and Semantics

- Multiple transport mechanisms in the order of preference
  - `a=tcap:SAVP/RTP AVP/RTP`

- Referring to multiple attributes
  - `a=pcfg:t=1 a=1,3,4,5,6,8`

- Alternatives in potential configurations
  - `a=pcfg:t=3|4 a=1|2`

- Optional capabilities
  - `a=pcfg:t=1 a=1,[2],3`

- Inheritance: all attributes specified per `m=` line without `[at]cap`
  - Become part of all potential and actual configurations of this media stream

Capability Negotiation Status

- To become RFC shortly
  - In Working Group Last Call (WGLC) right now.

- Coverage
  - Basic negotiation mechanisms
  - Essential feature set for alternative transports a basic parameters
  - Particularly security

- Complementary specifications
  - Media attribute sets for capability specifications
    - Do not want to inherit all the baggage from SDP
  - Discussion of further capability representation mechanisms
    - So far, all attributes are additive (to the basic attribute set)
    - Deleting or replacing attributes?
    - Syntax and interpretation are easy; generation is hard.