PUBLIC

IST R&D. FP6-Priority 2.
SPECIFIC TARGETED RESEARCH PROJECT
Project Deliverable

SUIT Doc Number	SUIT_265
Project Number	IST-4-028042
Project Acronym+Title	SUIT- Scalable, Ultra-fast and Interoperable Interactive Television
Deliverable Nature	Report
Deliverable Number	D4.2
Contractual Delivery Date	M12 (31 January 2007)
Actual Delivery Date	8 February 2007
Title of Deliverable	Session and Description Protocols
Contributing Workpackage	WP4
Project Starting Date; Duration	01/02/2006; 27 months
Dissemination Level	PU
Author(s)	Kenneth Vermeirsch (IBBT), Víctor Domingo Reguant (URL), Francesc Pinyol (URL), Francesc Enrich (URL), Michael Probst (IRT)

Abstract

This document presents detailed information about the session, description, and feedback protocols relating to the SUIT project. Session protocols allow terminals to discover available broadcast programmes and to request additional unicast material. Description protocols play a role in the adaptation of media to the terminal capabilities and network conditions by offering a precise description of the usage environment and its constraints. Feedback information allows the playout to adapt video streams to the networks and terminals, thereby optimizing resource expenditure.

Keyword list: Session protocols, session management, description protocols, MPEG-21, media adaptation, feedback, return channel, SDP, SD&S, AdaptationQoS, SOAP

PUBLIC

Session and Description Protocols

SUIT-265

08-02-2007

Table of Contents

1 Introduction

1.1 Scope

This deliverable is an output document of Activity 4.2, which deals with resource and session management throughout the delivery chain. The scope of this document is to describe the Session and Description Protocols that will be used in the project. These protocols serve as the basis for building an intelligent system that is capable of delivering, in a QoS-optimized way, the streaming media to SUIT gateways and terminals.

1.2 Objective

The objective of this deliverable is to give a detailed description of the session and description protocols that can be used in the SUIT project and demonstrator.

Session descriptions serve the purpose of informing terminals about which media providers are currently accessible, what they offer, and where this media can be found. Session protocols subsequently allow terminals to join or leave multicast sessions; others assist in creating and maintaining a new unicast session when a user wants to access media-on-demand.

Description protocols give a precise view of the usage environment in which multimedia will be consumed, such as information about the user's preferences, the terminal's capabilities, the network condition, and so on. The purpose of this information is to enable and assist in the intelligent adaptation of the media to this usage context. For example, based on the knowledge of the usage environment retrieved from a description protocol, media resources can be adapted to the terminal being served (e.g., scaled down to a spatial resolution the terminal can handle) or to the current network conditions (e.g., adapted to the available bandwidth so as to avoid network congestion).

1.3 Overview of A4.2 Activities

Within this deliverable, three requirements of the overall SUIT architecture are addressed:

1. The need to describe which TV channels (programmes) are available on the network. This calls for the adoption of a session description protocol, and will be dealt with in Chapter 2.
2. The need to announce this information to the SUIT terminals, so that after being switched on, a terminal can quickly find out what the available programmes are, and can access their descriptions. This requires the adoption of session announcement and management protocols, and is dealt with in Chapter 3.
3. A mechanism to give feedback to the playout, in order to allow intelligent playout-side video adaptation of unicast streams. This requires a format to describe said feedback information and a protocol to transport this information back to the playout. Furthermore, it is necessary to define the manner in which this information will be used by the playout in order to adapt bitstreams. Chapter 4 will deal with these matters.

2 Session and Services Description

A multimedia session is a set of multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session. A session description should convey enough information to decide whether or not to participate in a session.

This chapter discusses various protocols for session description that were considered for use in SUIT. In Chapter 5, a choice of protocols will be made based upon their merits and drawbacks.

2.1 The Session Description Protocol (SDP)

2.1.1 Introduction

When initiating multimedia teleconferences, voice-over-IP calls, streaming video, or other sessions, there is a requirement to convey media details, transport addresses, and other session description metadata to the participants.

SDP [rfc4566] provides a standard representation for such information, irrespective of how that information is transported. SDP is purely a format for session description -- it does not incorporate a transport protocol, and it is intended to use different transport protocols as appropriate, including the Session Announcement Protocol [rfc2974], Session Initiation Protocol [rfc3261], Real Time Streaming Protocol [rfc2326], electronic mail using the MIME extensions, and the Hypertext Transport Protocol.

The purpose of SDP is to convey information about media streams in multimedia sessions, in order to allow the recipients of a session description to participate in the session. A multimedia session, for these purposes, is defined as a set of media streams that exist for some duration of time.

Thus SDP information includes:

- Session name and purpose
- Time(s) the session is active
- The media comprising the session
- Information to receive those media (addresses, ports, formats and so on)

As resources necessary to participate in a session may be limited, some additional information may also be desirable:

- Information about the bandwidth to be used by the session
- Contact information for the person responsible for the session

In general, SDP must convey sufficient information to enable a terminal to join a session.

2.1.2 Session Description Format

An SDP session description consists of a number of lines of text of the form: <type>=<value> where <type> must be exactly one case-significant character and <value> is structured text whose format depends on <type>. In general, <value> is either a number of fields delimited by a single space character or a free format string, and is case-significant unless a specific field defines otherwise.

SDP session descriptions are entirely textual using the ISO 10646 character set in UTF-8 encoding. SDP field names and attributes names use only the US-ASCII subset of UTF-8, but textual fields and attribute values may use the full ISO 10646 character set.

Field and attribute values that use the full UTF-8 character set are never directly compared, hence there is no requirement for UTF-8 normalisation. The textual form, as opposed to a binary encoding such as ASN.1 or XDR, was chosen to enhance portability, to enable a variety of transports to be used, and to allow flexible, text-based toolkits to be used to generate and process session descriptions.

An SDP session description consists of a session-level section followed by zero or more media-level sections. The session-level part starts with a "v=" line and continues to the first media-level section. Each media-level section starts with an "m=" line and continues to the next media-level section or end of the whole session description. In general, session-level values are the default for all media unless overridden by an equivalent media-level value. Some lines in each description are required and some are optional,
but all must appear in exactly the order given here (the fixed order greatly enhances error detection and allows for a simple parser). Optional items are marked with a "*".

Session description
v= (protocol version)
o= (owner/creator and session identifier).
s= (session name)
i=* (session information)
u=* (URI of description)
e=* (email address)
p=* (phone number)
c=* (connection information - not required if included in all media)
b=* (bandwidth information)
One or more time descriptions (see below)
z=* (time zone adjustments)
k=* (encryption key)
a=* (zero or more session attribute lines)
Zero or more media descriptions (see below)

Time description
t= (time the session is active)
r=* (zero or more repeat times)

The connection ("c=") and attribute ("a=") information in the session-level section applies to all the media of that session unless overridden by connection information or an attribute of the same name in the media description.

A Media Description section includes

- The type of media (video, audio, etc)
- The transport protocol (RTP/UDP/IP, H.320, etc)
- The format of the media (H.261 video, H.264 video, MPEG video, etc)

For an IP multicast session, the following are also conveyed:

- Multicast address for media
- Transport Port for media

This address and port is the destination address and destination port of the multicast stream, whether it is sent, received, or both.

For an IP unicast session, the following are conveyed:

- Remote address for media
- Transport port for contact address

The semantics of this address and port depend on the media and transport protocol defined. By default, this is the remote address and remote port to which data is sent, and the remote address and local port on which to receive data. However, some media may define to use these to establish a control channel for the actual media flow.

Media description
- m= (media name and transport address)
- i=* (media title)
- c=* (connection information - optional if included at session-level)
- b=* (bandwidth information)
- k=* (encryption key)
- a=* (zero or more media attribute lines)

2.1.3 Detailed Description of Session Parameters

Table 1. Description of Session Parameters

Parameter	m/o	Name	Meaning	Format
a	o	Attributes	Additional properties (SDP-extension)	a=<attribute> a=<attribute>:<value>
b	o	Bandwidth	Proposed bandwidth to be used by the session or media.	b=<bwtype>:<bandwidth>
c	o	Connection Data	Information about network connection	c=<nettype><addrtype> <connection-address>
e	o	Email Address	Email address of the owner	e=<email-address>
p	o	Phone Number	Telephone number of the owner	p=<phone-number>
i	o	Session Information	Textual information about the session	i=<session description>
k	o	Encryption Key	Security key for media streams	k=<method> k=<method>:<encryption key>
m	m	Media	Name and address of the media stream	m=<media> <port> <proto> <fmt> ...
o	m	Origin	Originator of a session (username + origin's host address) plus a session identifier and version number	o=<username> <sess-id> <sess-version> <nettype> <addrtype> <unicast-address>
r	o	Repeat Times	Specify repeat times for a session	r=<repeat interval> <active duration> <offsets from start-time>
s	m	Session Name	Session name	s=<session name>
t	m	Time	Session duration	t=<start-time> <stop-time>
u	o	URI	Pointer to additional information about the session	u=<uri>
v	m	Version	Version of the used protocol	v=0
z	o	Time Zone Adjustment	Allows the sender to specify a list of these adjustment times and offsets from the base time.	z=<adjustment time> <offset> <adjustment time> <offset>...

Table 2. Detailed Description of Session Parameters

Name	Tags Description
Attributes **a=<attribute>** **a=<attribute>:<value>**	Attributes may be defined to be used as "session-level" attributes, "media-level" attributes, or both. Attribute fields may be of two forms.
Bandwidth **b=<bwtype>:<bandwidth>**	The <bwtype> is an alphanumeric modifier specifying the semantics of the <bandwidth> parameter. Two arguments are defined in the RFC: **CT** gives a total bandwidth figure for all the media at all sites. When using the CT modifier with RTP, if several RTP sessions are part of the conference, the conference total refers to total bandwidth of all RTP sessions. **AS** gives a bandwidth figure for a single media at a single site. For RTP-based applications [rfc3550], AS gives the RTP "session bandwidth". The <bandwidth> parameter is interpreted as kilobits per second by default.

Connection Data
 	
c=<nettype><addrtype>
<connection-address>
	A session description must contain either at least one "c=" field in each media description or a single "c=" field at the session level.
	<**nettype**> is the network type, which is a text string giving the type of network. Internet is specified by "IN".
	<addrtype> is the address type. This allows SDP to be used for sessions that are not IP based. We are using "IP4" or "IP6".
	<connection-address> is the connection address. Optional sub-fields may be appended to the connection address depending on the value of the <addrtype> field.
	
	When <addrtype> is IP4 or IP6, the connection address is defined as follows:
	
	• Sessions using an IPv4 multicast connection address must have a time to live (TTL) value present in addition to the multicast address. The TTL and the address together define the scope with which multicast packets in this conference will be sent. An example is `c=IN IP4 224.2.36.42/127`, indicating a TTL of 127.
	• Hierarchical or layered encoding schemes are data streams where the encoding from a single media source is split into a number of layers. The receiver can choose the desired quality (and hence bandwidth) by only subscribing to a subset of these layers. Such layered encodings are normally transmitted in multiple multicast groups to allow multicast pruning. This technique keeps unwanted traffic from sites only requiring certain hierarchical levels. For applications requiring multiple multicast groups, we allow the following notation to be used for the connection address:
	
	<base multicast address>[/<ttl>]/<number of addresses>
	
	If the number of addresses is not given, it is assumed to be one. Multicast addresses thus assigned are contiguously allocated starting with the base address, so that, for example: `c=IN IP4 224.2.1.1/127/3` would state that addresses 224.2.1.1, 224.2.1.2, and 224.2.1.3 are to be used, with a TTL of 127. This is semantically equivalent to including multiple "c=" lines in a media description:
	
	`c=IN IP4 224.2.1.1/127`
	`c=IN IP4 224.2.1.2/127`
	`c=IN IP4 224.2.1.3/127`
	
	Multiple addresses or "c=" lines may be specified on a per-media basis only if they provide multicast addresses for different layers in a hierarchical or layered encoding scheme.

Session Information

i=<session description>

One session level "i=" field per session description, and at most one "i=" field per media. If the "a=charset" attribute is present, it specifies the character set used in the "i=" field. If the "a=charset" attribute is not present, the "i=" field must contain ISO 10646 characters in UTF-8 encoding.

Media

m=<media> <port> <proto> <fmt> ...

A session description may contain a number of media descriptions. Each media description starts with an "m=" field and is terminated by either the next "m=" field or by the end of the session description.
A media field has several sub-fields:

<media> is the media type. Currently defined media are "audio", "video", "text", "application", and "message".

<port> is the L4 port to which the media stream is sent.

For applications where hierarchically encoded streams are being sent to a unicast address, it may be necessary to specify multiple ports. This is done using a similar notation to that used for IP multicast addresses in the "c=" field:

m=<media> <port>/<number of ports> <proto> <fmt> ...

For RTP, the default is that only the even-numbered ports are used for data with the corresponding one-higher odd ports used for the RTCP [rfc3550] traffic belonging to the RTP session, and the <number of ports> denoting the number of RTP sessions. For example:

m=video 49170/2 RTP/AVP 31

specifies that ports 49170 and 49171 form one RTP/RTCP pair and ports 49172 and 49173 form the second RTP/RTCP pair. RTP/AVP is the transport protocol and 31 is the format (see below).

If multiple addresses are specified in the "c=" field and multiple ports are specified in the "m=" field, a one-to-one mapping from port to the corresponding address is implied. For example:

c=IN IP4 224.2.1.1/127/2
m=video 49170/2 RTP/AVP 31

implies that address 224.2.1.1 is used with ports 49170 and 49171, and address 224.2.1.2 is used with ports 49172 and 49173.

<proto> is the transport protocol. The meaning of the transport protocol is dependent on the address type field in the relevant "c=" field. Thus a "c=" field of IP4 indicates that the transport protocol runs on top of IPv4. The following transport protocols are defined:

- udp: denotes an unspecified protocol running over UDP.
- RTP/AVP: denotes RTP used under the RTP Profile for Audio and Video Conferences with Minimal Control [rfc3550] running on top of UDP.
- RTP/SAVP: denotes the Secure Real-time Transport Protocol [rfc3711] running on top of UDP.

<fmt> is a media format description. The interpretation of the media format depends on the value of the <proto> sub-field.

- If the <proto> sub-field is "RTP/AVP" or "RTP/SAVP", the <fmt> sub-fields contain RTP payload type numbers. For dynamic payload type assignments, the "a=rtpmap:" attribute should be used to map from an RTP payload type number to a media encoding name that identifies the payload

	format. The "a=fmtp:" attribute may be used to specify format parameters.
Origin **o=<username> <sess-id> <sess-version> <nettype> <addrtype> <unicast-address>**	**<username>** is the user's login on the originating host. **<sess-id>** is a numeric string. In combination with the other parameters, it forms a globally unique identifier for the session. <**sess-version**> is a version number for this session description. <**nettype**> is a text string defining the type of network. Internet is specified by "IN". <**addrtype**> is a text string specifying the type of the address that follows. **<unicast-address>** is the address of the machine from which the session was created.
Session Name **s=<session name>**	There must be exactly one "s=" field per session description.
Time **t=<start-time> <stop-time>**	The "t=" lines specify the start and stop times for a session. These values are the decimal representation of Network Time Protocol (NTP) time values in seconds since 1900 [rfc1305]. Permanent sessions may be shown to the user as never being active unless there are associated repeat times that state precisely when the session will be active.
Version **v=**	v=0

2.1.4 Detailed Description of SDP Attributes

Table 3. Detailed Description of SDP Attributes

Attribute	Definition	level
a=cat:<category>	Category of the session. This is to enable a receiver to filter unwanted sessions by category.	Session
a=keywds:<keywords>	Identifies desired sessions at the receiver. This allows a receiver to select interesting session based on keywords describing the purpose of the session.	Session
a=tool:<name and version of tool>	This gives the name and version number of the tool used to create the session description.	Session
a=ptime:<packet time>	This gives the length of time in milliseconds represented by the media in a packet. This is probably only meaningful for audio data.	Media
a=maxptime:<maximum packet time>	This gives the maximum amount of media that can be encapsulated in each packet, expressed as time in milliseconds. This attribute is probably only meaningful for audio data.	Media
a = rtpmap: <payload type> <encoding name>/<clock rate> [/<encoding parameters>]	This attribute maps from an RTP payload type number (as used in an "m=" line) to an encoding name denoting the payload format to be used. It also provides information about the clock rate and encoding parameters. For audio streams, <encoding parameters> indicates the number of audio channels. For video streams, no encoding parameters are currently specified.	Media
a = recvonly	This specifies that the tools should be started in receive-only mode where applicable. Note that recvonly applies to the media only, not to any associated control protocol (RTCP).	Session/Media
a = sendrecv	This specifies that the tools should be started in send and receive mode. This is necessary for interactive conferences with tools that default to receive-only mode. If none of the "sendonly", "recvonly", "inactive", or "sendrecv" attributes is present, "sendrecv" must be assumed as the default for sessions that are not of the conference type "broadcast" or "H332".	Session/Media

a = sendonly	This specifies that the tools should be started in send-only mode. An example may be where different unicast addresses are to be used for forward and return traffic. In such a case, two media descriptions may be used, one marked sendonly and the other recvonly. Note that sendonly applies only to the media; any associated control protocol (e.g., RTCP) should still be received and processed as normal.	Session/Media
a=inactive	This specifies that the tools should be started in inactive mode. This is necessary for interactive conferences where users can put other users on hold. No media is sent over an inactive media stream. Note that an RTP-based system should still send RTCP, even if started inactive.	Session/Media
a=orient:<orientation>	Normally this is only used for a whiteboard or presentation tool. It specifies the orientation of the workspace on the screen. Permitted values are "portrait", "landscape", and "seascape" (upside-down landscape).	Media
a=type:<conference type>	This specifies the type of the conference. Suggested values are "broadcast", "meeting", "moderated", "test", and "H332". "recvonly" is the default for "type:broadcast" sessions.	Session
a=charset:<character set>	This specifies the character set to be used to display the session name and information data. By default, the ISO-10646 character set in UTF-8 encoding is used. If a more compact representation is required, other character sets may be used. For example, the ISO 8859-1 charset is specified with the following SDP attribute: `a=charset:ISO-8859-1`	Session
a=lang:<language tag>	As a session-level attribute, it specifies the default language for the session being described. As a media-level attribute, it specifies the language for that media, overriding any session-level language specified.	Session/Media
a=sdplang:<language tag>	As a session-level attribute, it specifies the language for the session description. As a media-level attribute, it specifies the language for any media-level SDP information field associated with that media.	Session/Media
a=quality:<quality>	This gives a suggestion for the quality of the encoding as an integer value. The intention of the quality attribute for video is to specify a non-default trade-off between frame-rate and still-image quality. For video, the value is in the range of 0 (worst) to 10 (best image quality).	Media
a = framerate:<frame rate>	This gives the maximum video frame rate in frames/sec. It is intended as a recommendation for the encoding of video data. Decimal representations of fractional values using the notation "<integer>.<fraction>" are allowed.	Media
a=fmtp:<format> <format specific parameters>	This attribute allows parameters that are specific to a particular format to be conveyed in a way that SDP does not have to understand them. The format must be one of the formats specified for the media. Format-specific parameters may be any set of parameters required to be conveyed by SDP and passed to the media tool that will use this format unchanged. At most one instance of this attribute is allowed for each format.	Media

2.1.5 Extensions to Describe MDC and SVC Media

An SDP session description may contain one or more media descriptions, each identifying a single media stream. A media description is identified by one "m=" line. If more than one "m=" line exist, indicating the same media type, a receiver cannot identify a specific relationship between those media.

An IETF draft, called draft-schierl-mmusic-layered-codec [schierl], exists which extends the SDP specification to support signalling of relationships between media. It enables signalling decoding dependency of different media descriptions with the same media type in SDP. This is required, for example, if media data is separated and transported in different network streams as a result of the use of a layered media coding process. Different reasons can be envisioned, for example the transporting of bitstream partitions of a hierarchical media coding process (also known as layered media coding process) or of a multi description coding (MDC) in different network streams. There is a strong relationship with the SVC draft [wenger].

The draft defines a new grouping type "DDP" (decoding dependency), to be used in conjunction with [rfc3388]. In addition, an attribute is specified describing the relationship of the media streams in a "DDP" group.

There may be various reasons for the concurrent transport of various media (as identified by a media description) of the same media type, among which certain dependencies may exist. But the basic idea for all cases is the separation of partitions of a media bitstream to allow scalability in network elements.

Two types of dependency are discussed in the following in more detail, as they are conceptually well understood:

- **Layered/hierarchical decoding dependencies:** In layered coding, the partitions of a media bitstream are known as media layers or simply layers. One or more layers may be transported in different network streams. A classic use case is known as receiver-driven layered multicast, in which a receiver selects a combination of media streams conveyed in their own (in this case RTP-) session in response to quality or bit-rate requirements.
  That is, each so-called enhancement layer referred to exactly one layer "below". The single exception has been the base layer, which is self-contained. Therefore, an identification of one enhancement layer fully specifies the operation point of a layered decoding scheme, including knowledge about all the other layers that need to be decoded.

- **Multiple description decoding dependencies**: In the most basic form of multiple descriptive coding (MDC), each partition forms an independent representation of the media. That is, decoding of any of the partition yields useful reproduced media data. When more than one partition is available, then a decoder can process them jointly, and the resulting media quality increases. The highest reproduced quality is available if all original partitions are available for decoding.

2.1.5.1 Use cases for layered and MDC coding and transport

- ***Receiver driven layered multicast.***

This technology is discussed in [RFC3550] and references therein. A receiver selects a combination of media streams conveyed in their own (in this case RTP-) session in response to quality or bit-rate requirements.

- ***Multiple end-to-end transmission with different properties***

Assume a unicast (point-to-point) topology, wherein one endpoint sends media to another. Assume further that different forms of media transmission are available. The difference may lie in the cost of the transmission (free, charged), in the available protection (unprotected/secure), in the quality of service

(guaranteed quality / best effort) or other factors. Layered and MDC coding allows to match the media characteristics to the available transmission path. For example, in layered coding it makes sense to convey the base layer over high QoS and/or over an encrypted transmission path. Enhancement layers, on the other hand, can be conveyed over best effort, as they are "optional" in their characteristic -- nice to have, but non-essential for media consumption. Similarly, while it is essential that the base layer is encrypted, there is (at least conceptually) no need to encrypt the enhancement layer, as the enhancement layer is meaningless without the base layer. In a different scenario, the base layer may be offered in a non-encrypted session as a free preview, and an encrypted enhancement layer allowing optimal quality play-back may only be accessible to users authorised by a conditional access mechanism.

2.1.5.2 Signalling media dependency (MDC and SVC) in SDP

The dependency signalling is only feasible between media descriptions described with a "m="-line and with an assigned media identification attribute ("mid").
The new grouping semantics Decoding Dependency "DDP" is defined as follows:

DDP associates a media stream, identified by its mid attribute, with a DDP group. Each media stream must be composed of an integer number of media partitions. All media streams of a DDP group must have the same type of coding dependency as signalled by the (Attribute for dependency signalling per media-stream and must belong to one media bitstream. All media streams must be part of at least one operation point. The DDP group type informs a receiver about the requirement for treating the media streams of the group according to the new media level attribute "depend".

Attribute for dependency signalling per media-stream

A new media-level value attribute is defined, "depend". The "identification-tag" (if used) is defined in [rfc3388]:

```
depend-attribute     = "a=depend:" dependency-type-tag
                       *(space identification-tag)
dependency-type-tag  = dependency
dependency           = "lay" / "mdc"
```

The "depend"-attribute describes the decoding dependency. The "depend"-attribute may be followed by a sequence of identification- tag(s) which identify the directly related media streams. The attribute may be used with multicast as well as with unicast transport addresses. The following types of dependencies are defined:

- **lay:** Layered decoding dependency. Identifies the described media stream as one or more partitions of a layered media bitstream. When lay is used, all media streams must be identified by the following identification-tag(s) that are required for a successful use of the media stream.
- **mdc:** Multi descriptive decoding dependency. Signals that the described media stream is one or more partitions of a multi description coding (MDC) media bitstream. Receiving more than one media stream of the group may enhance the decodable quality of the media bitstream. This type of dependency does not require the signalling of the depended media streams.

Examples

1. **Example for signalling transport of operation points of a layered video bitstream in different network streams:**

```
v=0
o=svcsrv 289083124 289083124 IN IP4
s=LAYERED VIDEO SIGNALING SUIT
t=0 0

c=IN IP4 224.2.17.12/127
a=group:DDP 1 2 3

m=video 40000 RTP/AVP 94
b=AS:96
a=framerate:15
a=rtpmap:94 h264/90000
a=mid:1

m=video 40002 RTP/AVP 95
b=AS:64
a=framerate:15
a=rtpmap:95 svc1/90000
a=mid:2
a=depend:lay 1

m=video 40004 RTP/AVP 96
b=AS:128
a=framerate:30
a=rtpmap:96 svc1/90000
a=mid:3
a=depend:lay 1
```

Base Layer: it is decodable by its own, because it has not the depend attribute

The enhancement layers depends on the base layer

2. **Example for signalling transport of streams of a multi description (MDC) video bitstream in different network streams:**

```
v=0
o=mdcsrv 289083124 289083124 IN IP4
s=MULTI DESCRIPTION VIDEO SIGNALING SUIT
t=0 0

a=group:DDP 1 2

m=video 40000 RTP/AVP 94
c=IN IP4 224.2.17.12/127
a=mid:1
a=depend:mdc

m=video 40002 RTP/AVP 95
c=IN IP4 224.2.17.13/127
a=mid:2
a=depend:mdc
```

2.1.6 Mapping MIME Parameters of RTP H.264/SVC Sessions in SDP Files

The optional MIME parameters specified in [rfc3984] apply. In addition the following optional MIME parameters apply:

sprop-scalability-info: This parameter may be used to convey the NAL unit containing the scalability information SEI message that must precede any other NAL units in decoding order. The parameter must not be used to indicate codec capability in any capability exchange procedure. The value of the parameter is the base64 representation of the NAL unit containing the scalability information SEI message as specified in [SVC]. In SUIT this NAL_SEI information is sent directly from the encoder in the RTP stream.

sprop-transport-priority: This parameter may be used to signal the transport priority indicator value(s) in terms of second and third bytes of the SVC NAL unit header for one or more SVC layer(s) conveyed in one RTP session. A transport priority indicator is base64 coded. If more than one layer is transmitted within one RTP session, the transport priority indicator value of each layer must be itemized with decreasing importance for decoding and must be comma-separated. Encoding considerations: This type is only defined for transfer via RTP (rfc 3550).

The MIME media type video/SVC string is mapped to fields in the Session Description Protocol (SDP) as follows:

- The media name in the "m=" line of SDP must be video.
- The encoding name in the "a=rtpmap" line of SDP must be SVC (the MIME subtype).
- The clock rate in the "a=rtpmap" line must be 90000.
- The optional parameters "profile-level-id", "max-mbps", "max-fs", "max-cpb", "max-dpb", "max-br", "redundant-pic-cap", "sprop-parameter-sets", "parameter-add", "packetization-mode", "sprop-interleaving-depth", "deint-buf-cap", "sprop-deint-buf-req",”sprop-init-buf-time", "sprop-max-don-diff", "max-rcmd-nalu-size", "sprop-transport-priority", and "sprop-scalability-info", when present, must be included in the "a=fmtp" line of SDP. These parameters are expressed as a MIME media type string, in the form of a semicolon separated list of parameter=value pairs.

Usually, the most common additional optional parameters used are:

-profile-level-id: If it is not present the baseline profile is used.
-sprop-parameters-sets: this parameter may be used to convey any sequence and picture parameter set NAL units that must precede any other NAL units in decoding order. In SUIT this information in conveyed in NAL_SPS and NAL_PPS, that are sent directly from the encoder in the RTP stream.
-packetization-mode: this parameter signals the properties of an RTP payload type or the capabilities of a receiver implementation. When the value of packetization-mode is equal to 0 or packetization-mode is not present, the single NAL mode must be used. When the value of packetization-mode is equal to 1, the non-interleaved mode, as defined in section 6.3 of [rfc3984], must be used. When the value of packetization-mode is equal to 2, the interleaved mode, as defined in section 6.4 of [rfc3984], must be used.

2.2 Session Description Protocol new generation (SDPng)

2.2.1 Introduction

SDPng [bormann], the IETF successor to SDP, is a radical departure from its predecessor. SDPng provides a common, XML-based application-independent framework for session description and capability description for different application scenarios. It allows for distributing "fixed" configuration descriptions in broadcast application scenarios but also supports the dynamic negotiation of session parameters for interactive multimedia conferences. In order to support a broad range of different applications, SDPng itself is completely application-agnostic. The base specification does not define any application-specific vocabulary, e.g., media types, codecs and their configuration parameters etc., but is intended as an extensible framework

that provides the necessary extensibility mechanisms for supporting both future applications and future transport and encoding mechanisms.

2.2.2 SDPng Design

SDPng provides fundamental mechanisms that support the inclusion of meta-information: descriptions of application components (media sessions) and alternative configurations can be labelled to enable later referencing, i.e., for associating meta-information. For example, in a multi-language TV broadcast session, the language information could be provided by a meta-information fragment that assigns language tags to media session descriptions by referencing them.

SDPng itself does not define vocabulary for specifying meta-information, but allows including arbitrary meta-information fragments, e.g., MPEG-7 descriptions. These descriptions may either be included inline or referenced by URIs.

An SDPng description is an XML document consisting of up to five parts:

- Capabilities: this section provides a list of individual capabilities. It is optional. The element type is called "cap".
- Definitions: this section provides definitions of commonly used parameters for later referencing. It is optional. The element type is called "def".
- Configurations: this section provides the description of the different conference components (applications in a conference). It must be present. The element type is called "cfg".
- Constraints: this section provides constraints on combinations of configurations. It is optional. The element type is called "constraints".
- Session Information: this section provides meta information on the conferences and on individual components. It is optional. The element type is called "info".

An SDPng description is an XML document. The document root must be an element of type "sdpng". The XML vocabulary for the SDPng base specification resides in the XML namespace http://www.iana.org/sdpng.

An Example of Capabilities SDPng element [bormann]:

```
<?xml version="1.0"?>
<sdpng xmlns="http://www.iana.org/sdpng"
      xmlns:sdpng="http://www.iana.org/sdpng">
  <cap>
     <video:codec name="h263+-enhanced">
     <video:encoding>H.263+</video:encoding>
     <video:resolution>QCIF</video:resolution>
     <video:framerate max="30"/>
     <h263plus:A>foo</h263plus:A>
     <h263plus:B>bar</h263plus:B>
    </video:codec>
    [...]
  </cap>
</sdpng>
```

The following example depicts a def element with one definition element of type "rtp:udp". This element is used to specify fixed parameters of an RTP session -- the allowable parameters would have been specified in a corresponding SDPng RTP package. A definition element is also presented and session information like an SDP file.

```
<?xml version="1.0"?>
<sdpng xmlns="http://www.iana.org/sdpng"
      xmlns:sdpng="http://www.iana.org/sdpng">
 <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:rtp-port>9456</rtp:rtp-port>
   <rtp:pt>1</rtp:pt>
  </rtp:udp>
 </def>
 <cfg>
  <component name="interactive-audio" media="audio" status="active">
   <alt name"alt1">
    <audio:codec ref="avp:pcmu"/>
    <rtp:udp ref="rtp-cfg1"/>
   </alt>
  </component>
 </cfg>
 <constraints>
  [...]
 </constraints>
<info>
  <part type="SDP">
   o=jdoe 2890844526 2890842807 IN IP4 10.47.16.5
   s=SDP SUIT
   i=Example
   t=2873397496 2873404696
  </part>
 </info>
</sdpng>
```

The SDPng standard defines a DTD and a schema that includes all the allowed parameters in each section of the SDPng file.

2.2.3 SDPng and SDP Comparison

After the overview of the SDPng and SDP protocols, a short comparison between the two protocols is presented in the following table:

Table 4: SDPng and SDP comparative

	SDPng	SDP
Document status	IETF Draft	RFC. RTP RFCs [rfc3550] [rfc3984] refer to it in order to signal RTP sessions and its associated parameters according the RTP payload.
	Independent application framework	dependent
Structure	XML based	Text based, key/value pairs
Extensibility	Easy. Native support for extensions	Complicated. Uses attributes concept for

		allowing extensions
Metadata format	Allows MPEG-7/MPEG-21 DIA descriptions	No support for MPEG-7/MPEG-21
Announcement	Supports SAP and RTSP transport	Supports SAP and RTSP transport
Usage	It is not commonly used	Commonly used
Support	No support in existing RTP libraries	Good support in existing RTP libraries
MDC/SVC support	Extensions for MDC and SVC are not defined	Draft of Extensions for MDC and SVC is defined
Session Info	Supports SDP session info	-

2.3 The MPEG-21 Digital Item Declaration Language

2.3.1 Introduction

The MPEG-21 Multimedia Framework (ISO/IEC 21000) is an interoperable framework for multimedia production, distribution and consumption. It aims to provide the 'big picture' in a highly automated multimedia delivery system, and it is designed to be independent of coding formats and network and terminal technologies. The MPEG-21 Multimedia Framework consists of many parts, a central one being the Digital Item Declaration standard (MPEG-21 DID).

MPEG-21 DID is all about describing multimedia assets. A Digital Item (DI) is the MPEG-21 representation of such an asset, and all transactions within MPEG-21 act upon DIs. MPEG-21 DID defines an XML-based format, with associated semantics provided as normative text, to describe DIs and the multimedia assets of which they are made up. This format is called the Digital Item Declaration Language (DIDL). The example DI below represents a video stream in the MPEG-21 framework:

MPEG-21 DIDL has a strong hierarchy. A Digital Item consists of a single or multiple elementary video/audio/image… streams, called Components. An Item can also contain nested Items. A Component, in turn, contains a Resource (which is the data itself, or a pointer to it) along with associated Descriptors. At a higher level, multiple Items can be grouped together into so-called Containers. Descriptors can be added in order to annotate certain elements. Figure 1 illustrates the DID hierarchy.

As an example, imagine a Compact Disc collection: the entire collection is a Container, in which each CD is an Item. Every CD Item may contain a Descriptor, containing the title, and several nested Items, representing the tracks of the CD, the back/front cover images, lyrics texts, etc. A track Item, in turn, could contain a Description with the track title and a Resource, pointing to the location of the MP3 file containing the track.

The DID document corresponding to this example would look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS">
 <Container id="cd_collection">
  <Descriptor>
   <Statement mimeType="text/plain">
    Compact Disc Collection
   </Statement>
  </Descriptor>
  <Item>
   <Descriptor>
    <Statement mimeType="text/plain">
     Eurovision 2000
    </Statement>
   </Descriptor>
   <Descriptor>
    <Component>
     <Resource mimeType="image/jpeg" ref="ftp://server/eurosong-cover.jpeg"/>
    </Component>
```

```
    </Descriptor>
    <Item id="track_01">
     <Descriptor>
      <Statement mimeType="text/plain">
       Fly On The Wings Of Love
      </Statement>
     </Descriptor>
     <Component>
      <Resource mimeType="audio/mpeg" ref="http://server/eurosong-2000-
01.mp3"/>
     </Component>
     <Component>
      <Resource mimeType="text/plain" ref="http://server/lyrics-0001.txt"/>
     </Component>
    </Item>
    <!-- ...other tracks... -->
   </Item>
   <!-- ...other albums... -->
 </Container>
</DIDL>
```

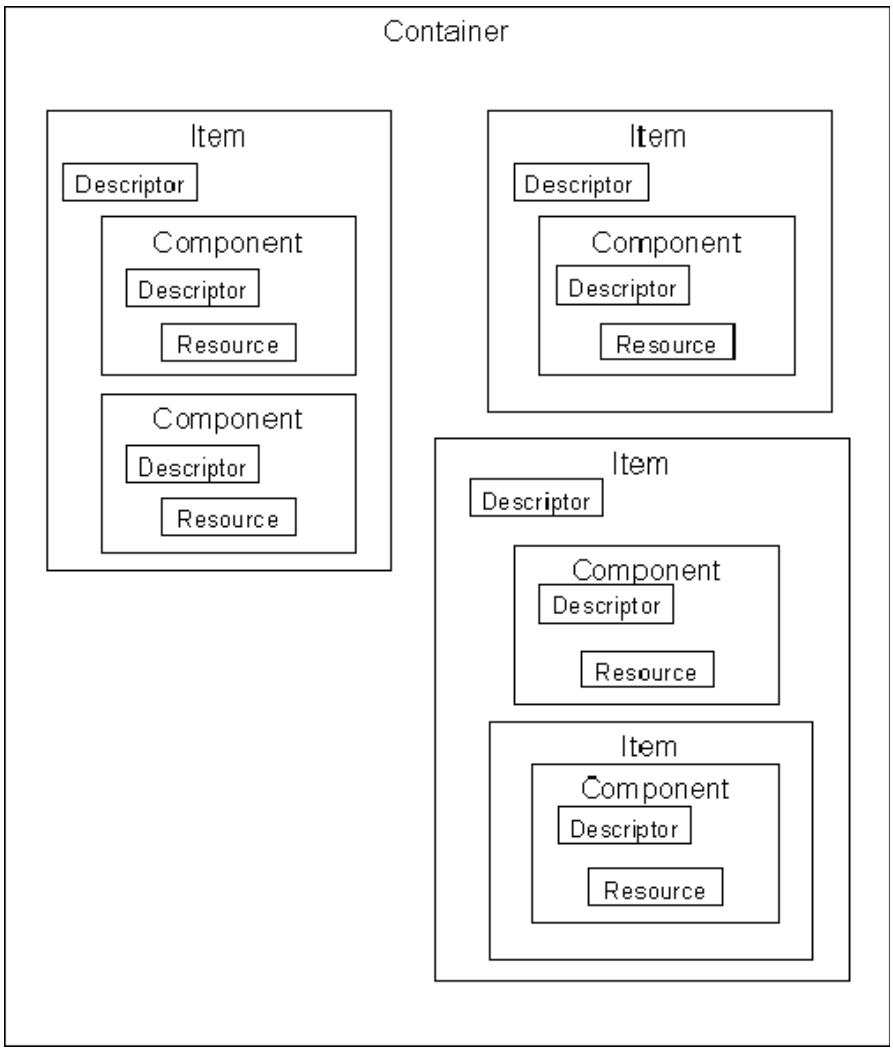


Figure 1. MPEG-21 DID hierarchy

2.3.2 Service Description

As the example from the previous section suggests, DIDL can be used to describe pre-recorded multimedia resources in video on demand scenarios. The entire offering of a service provider could be modelled as a Container. In a live broadcast scenario, a DID can be generated on the fly for streaming content. Since DIDL is a file format, not a protocol, it has the same role in the overall picture as an SDP file.

By extension DIDL could also be used to describe the available programmes on a TV network. The entire network can be seen as a collection of programmes, i.e., a Container. Each programme can be described as an Item, consisting of a number of media streams and some additional information, like the programme name, a logo, MHP applications, references to Teletext information, etc. A single DID document can contain all this information.

Inside a DID, the MPEG-7 metadata format can be used to describe certain aspects of the Digital Item. The relevant MPEG-7 descriptors are then simply included inline in the DID document, using XML namespaces to indicate that the descriptors are in the MPEG-7 format. For example, the coded bitrate of a video stream could be indicated by using MPEG-7, as in the following example fragment:

```
<Component>
 <Descriptor>
  <Statement mimeType=”text/xml”>
   <Mpeg7 xmlns=”urn:mpeg:mpeg7:schema:2001”>
    <Description xsi:type=”MediaDescriptionType”>
     <MediaInformation>
      <MediaProfile>
       <MediaFormat>
        <Content href=”http://content.playout.suit/on-demand-01.264”/>
        <BitRate>5000000</BitRate>
       </MediaFormat>
      </MediaProfile>
     </MediaInformation>
    </Description>
   </Mpeg7>
  </Statement>
 </Descriptor>
 <Resource mimeType=”video/mpeg” ref=”http://content.playout.suit/on-demand-
01.264”/>
</Component>
```

Where the SDP protocol is clearly very specifically targeted towards RTP-based applications, MPEG-21, as mentioned above, aims to be completely network-agnostic. This makes it somewhat more complicated in MPEG-21 DID to describe network-related parameters, like the IP address and UDP port number to be used for streaming. Two possibilities are:

1. Encoding the parameters inside the URI which points to the Resource;
2. Adding a Descriptor containing this information (e.g., in an application-specific way).

In any case, additional parsing will be required in order to obtain these parameters.

2.3.3 Description of MDC and Layered Streams

It can be beneficial to have a syntax to indicate dependencies (e.g., decoding dependencies) between streams. This functionality could be useful in layered coding, when different layers are stored in different bitstreams. Another application is MDC where there is no real decoding dependency between descriptions but where it is nonetheless useful to indicate that two streams are two MDC descriptions of a same video stream.

In MPEG-21 DIDL there is no explicit syntax to express these inter-stream dependencies. This is because in a Digital Item, each Component is assumed to be able to stand on its own. It is therefore not entirely correct

to, e.g., describe separate layers as separate Components (in the way each audio channel would be a separate Component), even when they are stored in separate bitstreams. However, dependency information could be emulated, e.g. by adding Descriptors which contain the dependency information. However, since this is not explicit syntax, this introduces an application-specific, non-interoperable notation.

2.3.4 Harmonising MPEG-21 with SDPng

It appears that there is substantial overlap between SDPng and MPEG-21 syntax elements; on the other hand many elements are present in one format but not in the other. Both formats have been developed independently without any interaction between IETF and ISO [m12002]. However, steps are being taken to harmonise both formats with the aim of making it easier to use SDPng to convey (fragments of) MPEG-21 information over the network. More information can be found in [m10996] and [m12002].

2.4 IPTV solutions: Service Discovery & Selection (SD & S)

2.4.1 Introduction

DVB-IPI (Digital Video Broadcast – Internet Protocol Infrastructure) deals with the delivery of DVB services over bi-directional IP networks for live broadcasts as well as for on-demand scenarios. As the convergence of networks to an all-IP infrastructure is an important theme in SUIT, the project is targeting DVB-IPI as one of the supported protocols.

Concerning session protocols, the DVB-IPI standard: ETSI TS 102 034 “Transport of MPEG-2 Based DVB Services over IP Based Networks” [ts102034] defines a mechanism called Service Discovery and Selection (SD&S) for embedding service-related information into a DVB-IPI service offering. More specifically, SD&S allows terminals to discover and then select the service providers, live broadcast offerings, on-demand offerings, and programme guides that are available on their DVB-IPI link. The service discovery information records are represented using the Extensible Mark-up Language (XML), and their Schemas are supplied in [ts102034] as a normative part of the standard.

2.4.2 Entry-Point Detection

With SD&S DVB-IPI defined a fully automated mechanism for standard IPTV clients to discover and monitor available services in a DVB-IPTV network. In the ideal case the user can immediately start watching TV without the need to manually scan the network for services.
The first step here is to find entry points, i.e. the locations where the client can access the service information. The detection of entry points uses standard protocols, well known from the internet. An entry point can be found by a client with one of the following methods (using this sequence):

1. A combination of DHCP with DNS
   The client’s IP settings may be configured using DHCP by the service provider. DHCP option 15 allows to provide the client with domain names. Together with the service name **_dvbservdsc** and the protocol (udp or tcp) the client can construct a DNS lookup for the SD&S entry point.
   Example: The SUIT project is a service provider and **suit.org** is the domain name which is owned by SUIT. **suit.org** identifies SUIT as a service provider in a DVB IPTV network. This domain name is provided via DHCP option 15 to the client. The client will assemble two DNS lookups:

   - _dvbservdsc._udp.suit.org
   - _dvbservdsc._tcp.suit.org

   If the lookup resolves into an IP address the client found an entry point. With udp or tcp the client specifies the delivery mode for SD&S. Please refer to section 2.4.4 for the two delivery alternatives.

2. DNS
   A simple modification of method 1. which uses **services.dvb.org** as the domain name. The lookup will be:

   - _dvbservdsc._udp.services.dvb.org
   - _dvbservdsc._tcp.services.dvb.org

3. DVB registered a multicast address with IANA for service discovery. It is 224.0.23.14. The client shall join the multicast group and it shall wait at least twice the maximum SD&S cycle time, which is in fact 1 minute, for SD&S sections.

4. If all of the methods above fail, the client shall ask the user to enter an entry point manually.

After getting the entry point the client starts the actual service discovery.

2.4.3 SD&S Records

The service discovery is a two step approach, firstly the client gets service providers and then it receives the service offering for each service provider. The format of the service provider and service offering information is defined by an XML schema in [TS102034].
SD&S records are delivered in segments. Segments are identified by a payload type, a segment id and a segment version.
The request to get a list of all service provider records with http is:

```
http://<entry_point>/dvb/sdns/sp_discovery?id=All
```

The following example shows a segment with payload id 1 containing a service provider record for the service provider SUIT. Inside the record the client finds the link to the service offerings of SUIT.

```
<?xml version="1.0" encoding="UTF-8"?>
<dvb:ServiceDiscovery xmlns:dvb="urn:dvb:ipi:sdns:2006>
<dvb:ServiceProviderDiscovery>
      <dvb:ServiceProvider DomainName="suit.org" Version="0">
            <dvb:Name Language="eng">SUIT</dvb:Name>
            <dvb:Description Language="eng">
                  The SUIT project offers Live TV ...
            </dvb:Description>
            <dvb:Offering>
                  <dvb:Pull Location="http://x.x.x.x:3937/dvb/sdns">
                        <dvb:PayloadId Id="2">
                              <dvb:Segment ID="0" Version="0"/>
                        </dvb:PayloadId>
                  </dvb:Pull>
                  <dvb:Push Port="3937" Address="224.0.23.14">
                        <dvb:PayloadId Id="5"/>
                  </dvb:Push>
            </dvb:Offering>
      </dvb:ServiceProvider>
</dvb:ServiceProviderDiscovery>
</dvb:ServiceDiscovery>
```

The service provider discovery record defines the service provider (and its existence) and contains the link to the actual service discovery records. In the example two different locations for the offerings are used, just to show that it is possible to mix unicast and multicast delivery. Segments with the payload type 2 (broadcast discovery) are delivered in unicast mode and segments with payload type 5 (package discovery) with multicast. For the unicast delivery all segments have to be listed in the record, because there is no other way for the client to know about a segment.
In the second step the client receives the service offerings. To stay with the example the client will get the BroadcastDiscoveryRecord with the http request:
http://x.x.x.x/dvb/sdns/service_discovery?id=suit.org&Payload=02&Segment=0000

The server returns a XML document that contains the broadcast discovery record with a list of services and their properties.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<dvb:ServiceDiscovery xmlns:dvb="urn:dvb:ipi:sdns:2006"
xmlns:tva="urn:tva:metadata:2005">
  <dvb:BroadcastDiscovery DomainName="suit.org">
    <dvb:ServiceList>
      <dvb:SingleService>
        <dvb:ServiceLocation>
          <dvb:IPMulticastAddress Port="5000" Address="230.1.2.3"/>
        </dvb:ServiceLocation>
        <dvb:TextualIdentifier ServiceName="suit_tv"/>
        <dvb:DVBTriplet ServiceId="40000" TSId="9999" OrigNetId="1"/>
          <dvb:SI ServiceType="16">
            <dvb:Name Language="eng">SUIT TV</dvb:Name>
          </dvb:SI>
          <dvb:VideoAttributes>
            <tva:Coding href="urn:dvb:ipdc:esg:cs:VideoCodecCS:9"></tva:Coding>
            <tva:Scan>interlaced</tva:Scan>
            <tva:HorizontalSize>720</tva:HorizontalSize>
            <tva:VerticalSize>576</tva:VerticalSize>
            <tva:AspectRatio>16:9</tva:AspectRatio>
            <tva:Color type="color"></tva:Color>
            <tva:FrameRate>25</tva:FrameRate>
          </dvb:VideoAttributes>
      </dvb:SingleService>
    </dvb:ServiceList>
  </dvb:BroadcastDiscovery>
</dvb:ServiceDiscovery>
```

The example signals one service. The service is identified by a textual identifier which is **suit_tv.suit.org**. The actual service name for display to the user is “SUIT TV”. The service type is 0x16 (hex!), means a service using an advanced codec. The exact codec type is defined in the VideoAttributes element. Here the codec is MPEG4-AVC. The alternative here is VC-1. If the Coding element is omitted the video codec is MPEG2, but in that case also the service type has to be different: 0x1. The example shows all possible video attributes in the schema, like resolution and aspect ratio.
The service location element contains the link to the media stream which is either a multicast address or a RTSP URL. The example service is multicast.

2.4.4 SD&S Transport Protocols

For the multicast delivery SD&S records are encoded in STP sections and transmitted with the DVB STP protocol over UDP/IP multicast. HTTP is used for the unicast delivery method to request each single record from a SD&S server.

The protocols are defined in [TS102034]. A description can also be found in SUIT Deliverable 4.1.

2.4.5 Extensions for MHP

The DVB subgroup TM-TAM is working (January 2007) on MHP-IPTV – a specification to integrate DVB-MHP with DVB-IP. One aspect of MHP-IPTV is the signalling of MHP applications in SD&S. In classical DVB networks applications are signalled inside the transport stream in the so-called Application Information Table (AIT). MHP-IPTV includes an XML representation of the AIT – the XAIT. The specification foresees to embed the XAIT into the BroadcastDiscovery record if it signals a service bound application. Unbound applications are signalled in the ServiceProvider record.

The following example is based on the current draft. It is assumed that the ApplicationList element will be added by DVB-IPI inside the SingleService structure.

```
<dvb:ApplicationList>
<dvbmhp:application>
  <dvbmhp:appName Language="eng">SUIT MHP application</dvbmhp:appName>
  <dvbmhp:applicationIdentifier>
    <dvbmhp:orgId>65000</dvbmhp:orgId>
    <dvbmhp:appId>65000</dvbmhp:appId>
  </dvbmhp:applicationIdentifier>
  <dvbmhp:applicationDescriptor>
    <dvbmhp:type><dvbmhp:DvbApp>DVB-J</dvbmhp:DvbApp></dvbmhp:type>
    <dvbmhp:controlCode>AUTOSTART</dvbmhp:controlCode>
    <dvbmhp:visibility>NOT_VISIBLE_ALL</dvbmhp:visibility>
    <dvbmhp:serviceBound>true</dvbmhp:serviceBound>
    <dvbmhp:priority>0</dvbmhp:priority>
    <dvbmhp:version>0</dvbmhp:version>
    <dvbmhp:mhpVersion>
      <dvbmhp:profile>0</dvbmhp:profile>
      <dvbmhp:versionMajor>1</dvbmhp:versionMajor>
      <dvbmhp:versionMinor>0</dvbmhp:versionMinor>
      <dvbmhp:versionMicro>3</dvbmhp:versionMicro>
    </dvbmhp:mhpVersion>
  </dvbmhp:applicationDescriptor>
  <dvbmhp:applicationSpecificDescriptor>
    <dvbmhp:dvbjDescriptor>
      <dvbmhp:location>http://www.suit.org</dvbmhp:location>
      <dvbmhp:applicationStructure>
        <dvbmhp:classPathExtension>applications</dvbmhp:classPathExtension>
        <dvbmhp:initialClass>org.suit.SuitCoD.class</dvbmhp:initialClass>
      </dvbmhp:applicationStructure>
    </dvbmhp:dvbjDescriptor>
  </dvbmhp:applicationSpecificDescriptor>
</dvbmhp:application>
</dvb:ApplicationList>
```

2.4.6 Support of SDP

SUIT will add support for the Session Description Protocol in SD&S to allow the signalling of video streams which are described by a separate SDP document. The service location in SD&S will not link to the media stream directly but to the SDP description of the service.

Example:

```
<dvb:SingleService>
  <dvb:ServiceLocation>
    <suit:STPURL href="http://www.suit.org/sdp/suit_tv1.sdp"/>
  </dvb:ServiceLocation>
  <dvb:TextualIdentifier ServiceName="suit_tv"/>
  [...]
</dvb:SingleService>
```

3 Session Management Protocols

When descriptions of the available services and service providers have been created, these descriptions need to be made known to all active terminals on the network. Depending on the network technology being used, there may or may not already be a mechanism in place to take care of delivering this information. For example, in DVB-IPI described above, the SD&S mechanism already provides this functionality. A generic protocol for use on multicast-enabled IP networks is the Session Announcement Protocol (SAP) and is described in detail below.

Once a terminal has received sufficient information about the available services and service providers on the network, it can proceed to the selection of a service to view. This is usually done by presenting the list of available services to the user who then chooses which service they wish to view. Depending on the scenario under consideration, IGMP or RTSP may be more appropriate, with IGMP being the simpler of the two protocols, and RTSP the most flexible and feature rich. The SD&S mechanism of DVB-IPI also utilizes these two protocols as back-end protocols, so the SUIT terminal must be able to act as a RTSP client and should have IGMP support as well.

3.1 The Session Announcement Protocol (SAP)

In a multicast scenario the Session Announcement Protocol (SAPv2) (RFC2974) [rfc2974] can be used to announce the available programmes to the terminals. As an IETF standard, it has the advantage of not being tied to certain network technologies: it can be used on any multicast-enabled IP network. The following paragraphs offer a detailed, albeit not complete, description of SAP; for the full authoritative specification, the reader is referred to the relevant RFC [rfc2974]. SAP version 1 is deprecated and must not be used.

3.1.1 Design

SAP is stateless from the server's point of view, meaning clients are not supposed to respond to announcement packets. An announcement server cannot know how many terminals are listening for announcement messages at a given moment. Also, there are no guarantees for delivery beyond what is provided by standard UDP/IP.

Therefore, announcement packets are periodically multicasted to a well-known multicast address and with a well-defined maximum retransmission interval. The multicasting is done with the same scope as the announced multicast sessions. This allows configuration-less service discovery by the terminals.

SAP messages carry header information such as the originating source, an authentication header, message identifier hash and several flags. Besides the headers, SAP messages carry a payload describing the session it announces. The payload type is specified by including its mime-type in the SAP header and may be freely chosen according to the application's requirements. However, the use of SDP or SDPng is recommended by the RFC. In SUIT, the text/xml mime type could be used to transport MPEG-21 documents inside SAP messages.

3.1.2 Addressing

SAP announcements are sent on port 9875. The multicast address to send the announcements to can be derived as follows:

- For IPv4 global scope multicast sessions, which are in the range of 224.2.128.0 to 224.2.255.255, SAP announcements must be sent to the 224.2.127.254 multicast address.
- When using administered multicast scope (as defined in [rfc2365]) the announcements must be sent to the highest address of the relevant administrative scope zone. For example, if the scope range is 239.16.32.0 to 239.16.33.255, then 239.16.33.255 is used for SAP announcements.

- IPv6 multicast sessions are to be announced on FF0X:0:0:0:0:0:2:7FFE, where X is the 4-bit multicast scope value. For example, an announcement for a link-local multicast session using the address FF02:0:0:0:0:0:1234:5678 should be advertised on address FF02:0:0:0:0:0:2:7FFE.

3.1.3 Session Deletion and Modification

There are several ways in which a SAP announcement can expire:

- Explicit timeout: The session description payload contains session start and end time stamps. If the current time is later than the session end time stamp, the session should be considered expired and removed from the SAP listener's session cache.
- Implicit timeout: A receiver can estimate the announcement interval of a session based on previously received announcement messages. If no further messages are received for that service within ten times this period or within one hour (whichever is greater), then the session must be considered expired and deleted from the receiver's session cache.
- Explicit deletion: When a session deletion message is received, the relevant session must be deleted from the receiver's cache immediately. Session deletion packets without a valid and authentication header matching that of previous announcement packages must be ignored.

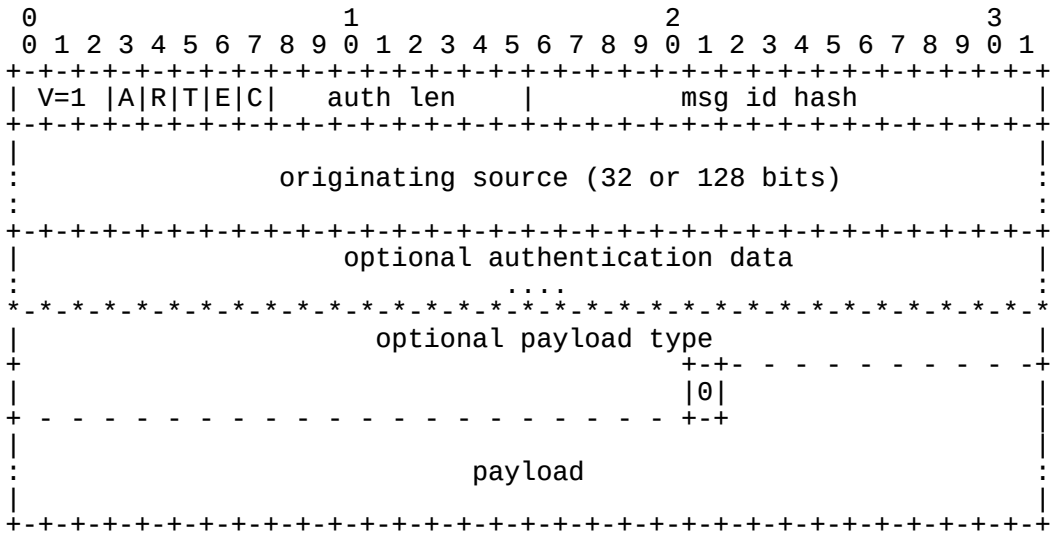
Session Modification

Updated information for existing sessions can be sent without first issuing a deletion of the existing session; however, the version hash in the headers must be changed to signal to the receivers that reparsing of the session description is necessary.

3.1.4 SAP Message Format

A SAP message is structured as depicted below. The semantics of the fields is as follows:

- **V**: Protocol Version Number. Must be equal to 1 in SAPv2.
- **A**: Address Type. Indicates whether the originating source field contains a 32-bit IPv4 address (A = 0) or a 128-bit IPv6 address (A = 1).
- **R**: Reserved. Must be 0. Must be ignored by receivers.
- **T**: Message Type. Indicates whether the packet contains a session announcement (T = 0) or a session deletion (T = 1) message.
- **E**: Encryption Flag. If set to 1, the payload is encrypted (see below).
- **C**: Compression Flag. If set to 1, the payload is compressed using the standard zlib compression algorithm. If both **E** and **C** are set to 1, the compression must be performed first.
- **Authentication Length**: This 8 bit unsigned int contains the number of 32 bit words containing authentication data. If this field equals 0, no authentication data is sent.
- **Message Identifier Hash**: Must be nonzero. This is a version identifier for the session information carried in the payload. This field must be chosen such that the combination of Originating Source and Message Identifier Hash is a globally unique identifier for a given version of a given session; in other words, any announcer must make sure that each session it announces has a Message Identifier Hash that is unique within that announcer, and that this hash is changed each time the session description is modified.
- **Originating Source**: This is the IPv4/IPv6 address (in network byte order) of the original source of the message.



3.1.5 Payload and Payload Type

Since SAP is intrinsically a one-way protocol without the possibility for payload type negotiation between announcer and listeners, the RFC expresses concern about possible proliferation of content-types; and therefore it recommends against using anything other than SDP as the payload type for session description. However, within the context of SUIT, we should consider overstepping this restriction, because of a number of reasons. Firstly, SAP is a relatively straightforward carrier mechanism for session description delivery. Secondly, it is well suited towards use in an all-IP environment. Thirdly, despite the recommendation only to use SDP, a payload type field has nonetheless been provided in the SAP header. This means that non SUIT-compliant terminals on the network will simply discard the SUIT-related session announcements. Thus, a case can be made to use SAP in conjunction with modern, structured session description formats such as SDPng or MPEG-21 DID.

3.2 The Internet Group Management Protocol (IGMP)

After the service discovery phase, a terminal has to select a service for viewing. This involves signalling to the playout and/or the network that it wishes to join a given session. For multicast services, the Internet Group Management Protocol (IGMP) [rfc2236], [rfc3376] is the native method for a terminal to join or leave the multicast group associated with the service.

IGMP is an integral part of the Internet Protocol (IPv4), and as such, should be considered a L3 protocol. IGMP's main purpose is to inform multicast-aware network nodes (routers) that packets belonging to a given multicast session should be forwarded to the terminal that issued the request.

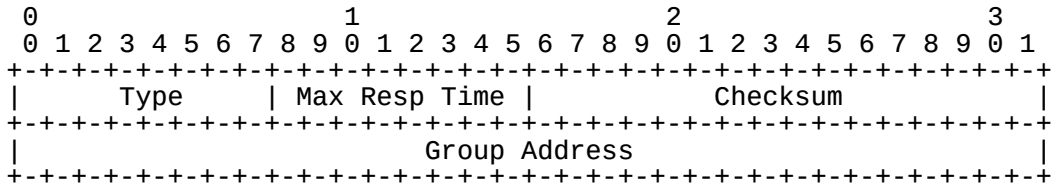
IGMP is only defined in cooperation with IPv4. In an IPv6 environment the Multicast Listener Discovery (MLD) protocol performs a similar function. It is not covered here.

The difference between IGMPv2 (RFC2236) and IGMPv3 (RFC3376) is that the latter adds provisioning for source filtering, which allows a terminal to filter multicast packets based on the sender of the packets. This mechanism is described in full in the RFC [rfc3376] and will not be covered here.

It is important to note that IGMP is a low-level, routing-related protocol. It was not designed to offer high-level functionality. As a consequence, the streaming server has little or no control about which terminals join which sessions, so this method is best suited for free-of-charge services. When more advanced functionality is desired, such as billing or collecting viewing statistics, the application-level (L5) Real Time Streaming Protocol is a better choice.

3.2.1 IGMP Message Format

IGMP messages are carried by IP datagrams, with an IP protocol number of 2. The IP header must include the Router Alert option and must have TTL equal to 1. The IGMP messages relevant to terminals have the following format:



The semantics of the fields is as follows:

- **Type**: Indicates the kind of message, as described in the next section.
- **Max Response Time**: This field is only meaningful in Membership Query messages (for details, the reader is referred to the RFC [rfc2236]. In all other message types, it should be set to zero by the sender and ignored by receivers.
- **Checksum**: This 16-bit field is a hash of the entire IGMP packet (i.e., the IP payload). It must be verified by the receiver before further processing takes place.
- **Group Address**: This field contains the IPv4 multicast address subject to the message. In message types for which this field has no meaning, it must be set to zero.

3.2.2 IGMP Message Types

The message type is indicated by an 8-bit field. Messages of unrecognized type should be silently ignored. New message types may be defined in future versions or for other uses. The standardized message types are given below:

- **0x11**: Membership Query. Issued by routers to query if there are any members on the network for a given group. If multiple routers in the same scope try to query, the one with the lowest IP address is elected as the sole querying router. When group members on the network receive the Membership Query, they start a random timer; when this timer expires, they reply with a Membership Report, informing the querying router that at least one member is listening to the multicast group. All other members receive this query too, and will stop their timer and return to the idle state. This is to avoid a packet storm of Membership Reports.
- **0x12**: Version 1 Membership Report (see below).
- **0x16**: Version 2 Membership Report (see below).
- **0x17**: Leave Group. Sent to the *all-routers* multicast group (224.0.0.2) by a member when it leaves the group. If there are other members left on the network, the member may choose not to send the Leave Group message, thereby saving bandwidth. This message allows routers to stop forwarding unneeded multicast packets sooner. Without the addition of this message, the router could only decide to stop forwarding certain multicast sessions after a Membership Query round, which only takes place every few minutes.
- **0x22**: Version 3 Membership Report (see below).

Membership Reports are sent in reply to a Query. Also, when a host wishes to join a multicast group which is not yet being routed to its network, it must immediately send an unsolicited Membership Report, to ask the routers to start forwarding this multicast session.

For more information about the timers in IGMP, we refer to the RFCs.

3.2.3 IGMP Implementations

Since IGMP is a Layer 3 protocol, it tends to be implemented as part of the OS kernel. Processes running on the machine can issue their desired membership by performing a specific system call on the network socket they opened for receiving the multicast packets. The kernel's TCP/IP stack will take care of the necessary filtering and IGMP messaging, and deliver the correct multicast packets to the correct sockets. From that point on, the application can simply read from the socket as if it were a regular datagram socket.

For example, in most Unix-like systems (and on Windows using "Winsock"), group membership is controlled by calling setsockopt(2) on the socket with the option names IP_ADD_MEMBERSHIP and IP_DROP_MEMBERSHIP. Multicast data can subsequently be sent and received using the familiar sendto(2) and recvfrom(2) system calls.

3.3 The Real Time Streaming Protocol (RTSP)

For the supervision of unicast services (media-on-demand) and of multicast sessions that require more functionality, an application-level protocol is a better choice. IETF's Real Time Streaming Protocol (RTSP) [rfc2326] provides this functionality.

3.3.1 Protocol Design

RTSP is a L5 protocol defining a command set for setting up and tearing down streaming sessions between a streaming server and a single client. The session is initiated by the client. RTSP commands also exist for controlling the delivery of the stream, resulting in VCR-like functionality, such as pausing and resuming, recording, and playback at different speeds. It allows a client to access files on the server at arbitrary time points, resulting in fast-forward/rewind functionality.

For multicast services, the use of RTSP has several advantages over IGMP: it works better through Network Address Translation (NAT) routers in the user's premises; it allows consumer identification for billing of pay-per-view material; and the number of viewers of a given service can be counted. Quite obviously, because of the multicast nature of the service, RTSP commands for stream control (like PAUSE) are not supported in this case.

Streamed media data is not transported over the RTSP connection but uses its own, separate connection. In most cases, RTP is used as the transport protocol for data. Both unicast and multicast data transport is supported.

The RTSP message format follows the HTTP style of header fields, method line, and body, allowing existing parsers to be reused. Furthermore, all HTTP authentication and proxying mechanisms can be reused in RTSP; and capability negotiation between server and client is possible. Many of the headers have been inherited from HTTP, such as `User-Agent`, `Accept`, `Content-Type`, `Content-Length`, and `Content-Encoding`. Others are specifically defined by RTSP, such as `Session`, `Transport`, and `Scale`. Important new headers will be discussed below. The full list of supported headers for each method can be found in [rfc2326].

However, unlike HTTP, RTSP is stateful, meaning a server must keep track of sessions and their associated state. An RTSP session is between a server and a single client, but can actually include multiple media streams at once, which will be streamed to the client in parallel. The term 'presentation' is used in RTSP for a collection of media streams sharing a single time axis (for example, a video stream and two audio streams in different languages).

3.3.2 Presentations and Streams

The structure of a presentation and the properties of the component media streams make up the presentation description. The client can use it to select the most appropriate media for its purpose. Each media stream is identified by an RTSP URL. The format of this description and the way in which it is to be retrieved by a client are outside of the scope of the RTSP specification. Hence, the presentation description could for example be formatted in SDP/SDPng or in MPEG-21, as discussed earlier in this document.

RTSP supports several playback time notations: SMPTE Relative Timestamps, Normal Play Time (NPT) and Absolute Time (ISO 8601 using UTC).

3.3.3 RTSP Methods for Transport Management

The following methods are defined in RTSP for managing the transport of the media streams between client and streaming servers:

- **SETUP**: Issued by a client to configure the transportation of a media stream. The request includes the stream's RTSP URL as well as the `Transport` header which contains the desired streaming protocol, delivery model (unicast or multicast), and port numbers. The server will generate a new session identifier for the stream and send it in its response to the client, using the `Session` header.

- **DESCRIBE** and **ANNOUNCE**: Used to update the service information for the running session. DESCRIBE is issued by the client when it wishes to have up-to-date information. ANNOUNCE can be sent asynchronously by the server (i.e., pushed to the client), to force clients to update their cached service information. (ANNOUNCE also has a second function: it can be sent by a client that wishes to publish media of its own. This functionality is important in teleconferencing, but not in video broadcasting).

- **TEARDOWN**: Stops the specified stream, allowing associated server-side resources to be freed.

- **REDIRECT**: Sent by the server to instruct the client to fetch the stream from elsewhere. Using REDIRECT messages, server-side load balancing is made possible, by having a 'master server' which redirects all requests to a pool of 'slave' streaming servers.

- **GET_PARAMETER**: Allows server or client to request certain parameter values from the peer. For example, in DVB-IPI, the following parameters are mandatory:

    - `Stream-state`: holds the current state of the RTSP session. Possible values are `playing`, `paused`, and `stopped`.

    - `position`: holds the current time position (in NPT format) in the stream, in the case of a media-on-demand session.

3.3.4 RTSP Methods and Headers for Streaming Control

Apart from managing media transport, RTSP also has provisioning to control the streaming servers in a (unicast) session. The methods defined for that purpose are:

- **PLAY**: Starts streaming part of the stream as specified by the range parameter (or lacking a range, the entire stream) or resumes streaming if the session was in the paused state.

- **PAUSE**: Suspends streaming until the PLAY command or, when a range parameter is supplied, for a certain amount of time starting at a certain moment in the future.

Besides these methods, several headers also carry information that influences the delivery of the streamed media:

- **Range**: Specifies a range on the temporal axis. When used in a PLAY request, only that range of the media will be streamed (Actual start and end point may differ slightly based upon codec requirements, e.g. video streaming can only start at the beginning of a GOP.) When used in a PAUSE request, it specifies when and for how long to pause.

- **Scale**: Specifies how fast the media should be streamed compared to wall clock time, thus allowing slow-motion playback or fast-forward/rewind at different speeds. For example, a value of 0.5 indicates half speed, while a value of -4 indicates rewinding at quadruple speed. Streaming servers capable of fast-forward/reverse should not diverge from the normal bandwidth too much, so they should not just send the original stream at a higher rate; instead, they should stream a scaled-down version, e.g. containing only key pictures. A server may only support a limited set of scale values; in that case, the server will perform rounding on the requested scale value and report the actual parameter it used in the response header.

- **Speed**: Requests that the data delivery rate of the stream be scaled up or down by the supplied factor. This allows for slow-motion/fast-forward without requiring adaptation of the stream, but the bandwidth of the stream will change accordingly, so care must be taken not to exhaust the available bandwidth.

4 Feedback Channel Protocols

In unicast scenarios, the return channel capability offered by the last-mile networks is exploited in order to create an intelligent playout system. Feedback information about the terminal and the network can be used to adapt the unicast video stream, and make it more suitable for the terminal in question, or to avoid network congestion.

In the return channel between gateway and playout, tools from the MPEG-21 Digital Item Adaptation (DIA) suite will be used. The return channel work flow can be broken down into these parts:

1. Feedback information, such as device information and network statistics, are collected by the terminal. This process is outside of the scope of this deliverable.
2. This feedback information is gathered in a standardized format, so that it can be understood by the playout. In SUIT, this format will conform to the MPEG-21 DIA-UED (Usage Environment Description) specification.
3. The document tailored in steps 1 and 2 is periodically sent to the playout via the return channel. In the SUIT demonstrator, the playout will expose a Web Service to which clients can connect and upload their DIA-UED documents.
4. In the playout, the received document is parsed and the feedback information is extracted. For this task, an MPEG-21 DIA parser is employed.
5. Using MPEG-21 AdaptationQoS, the feedback information is converted to the parameters needed to drive the bitstream extractor.

The following sections present an in-depth description of the protocols mentioned above, and of implementation choices and issues related to them.

4.1 Usage Environment Description: MPEG-21 DIA-UED

In order to support adaptation of media resources, a means is needed to accurately describe its usage context such as terminal capabilities and network characteristics. The Digital Item Adaptation part of the MPEG-21 Multimedia Framework provides tools to describe the usage context in an unambiguous and interoperable manner, as well as tools to specify the adaptation process itself.

4.1.1 Overview of MPEG-21 Digital Item Adaptation

MPEG-21 part 7: Digital Item Adaptation (DIA) specifies tools that can be used for the adaptation of Digital Items to network, storage or end-user constraints. Adaptation of Digital Items is required in order to achieve interoperable transparent access to multimedia resources by shielding Users from network and terminal installation, management and implementation issues. Figure 1 illustrates the conceptual DIA architecture. As can be seen, an adapted Digital Item is generated by subjecting an original Digital Item to a Digital Item Adaptation Engine. This engine modifies the Digital Item's description (its DID) as well as its multimedia resources; hence, the engine consists of two parts: a Resource Adaptation Engine and a Description Adaptation Engine.

It must be noted that the engines themselves are not normatively described by the standard. The scope of the standardization consists of the DIA descriptions and the format-independent mechanisms that are used to support Digital Item Adaptation in terms of Resource Adaptation, Description Adaptation and Quality of Service management. These descriptions and mechanisms are referred to in Figure 2 as DIA Tools.

Figure 3 illustrates the eight major categories of tools that together make up the MPEG-21 DIA standard. In the context of the SUIT project, the Usage Environment Description (UED) tool will be of primary importance.

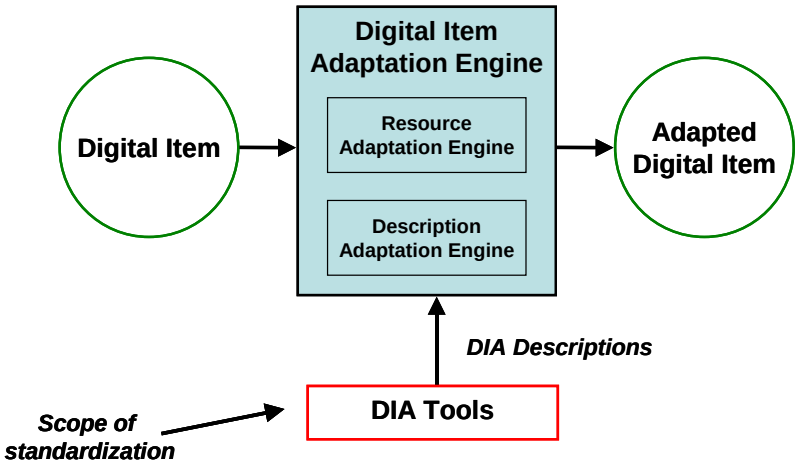


Figure 2. The Digital Item Adaptation Concept

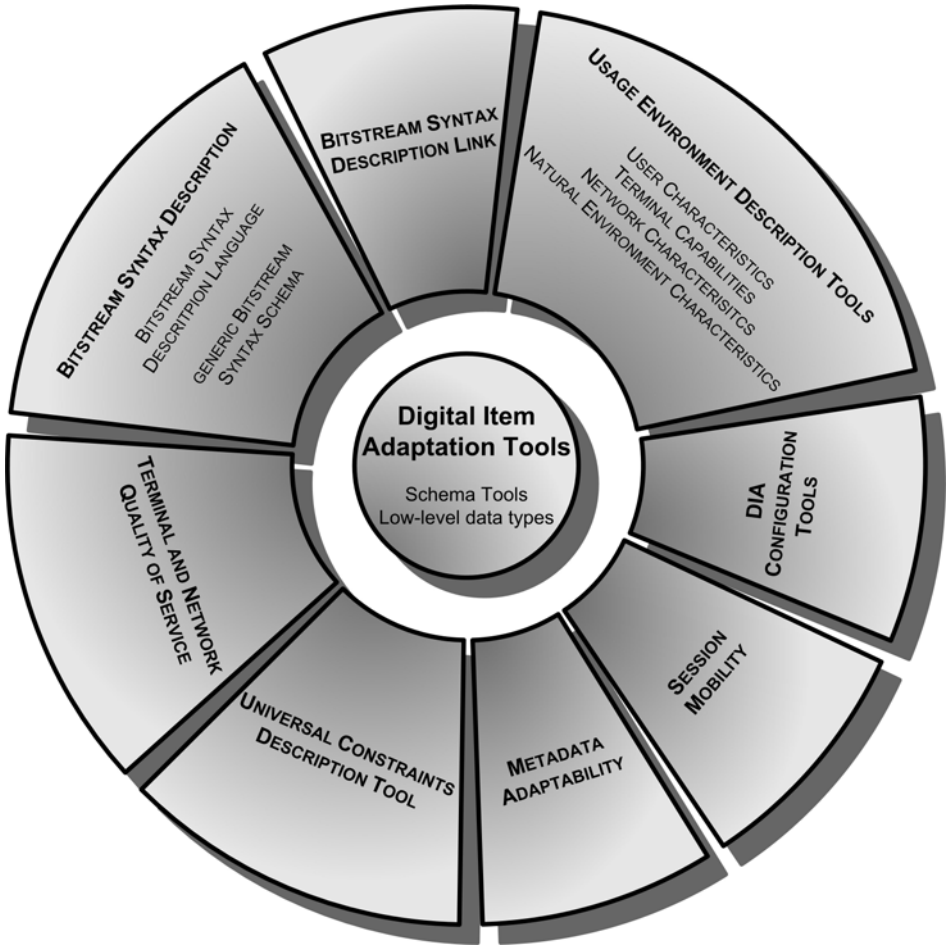


Figure 3. The Eight Categories of DIA Tools

4.1.2 MPEG-21 DIA Usage Environment Description (UED)

The UED tools provide descriptive information about the environment in which multimedia works are to be used. They can describe various aspects of the environment which can be used for Digital Item Adaptation, and are categorized as follows:

1. User characteristics: describe general User information, preferences, usage history, accessibility and mobility characteristics of the User, who is obviously a part of the consumption environment;

2. Terminal capabilities: in order to satisfy a terminal's processing and consumption constraints a wide variety of terminal capabilities and characteristics can be described, such as codec capabilities, power, storage and I/O characteristics, and display and audio output capabilities.

3. Network characteristics: describe network characteristics and conditions such as available bandwidth and delay and error characteristics.

4. Natural environment characteristics: describe usage aspects such as time and location of resource consumption, as well as the audiovisual consumption context such as current illumination characteristics and audio noise levels.

For the purpose of bitstream adaptation to suit the terminal capabilities as well as the network conditions, the SUIT project will mainly make use of the second and third categories from the list above.

The Usage Environment Description tool is specified in the form of an XML Schema every description must conform to. Hence, this schema specifies exactly what can be described by UED and can be used to validate whether a given XML document is a valid Usage Environment Description. A disadvantage of this is that the UED tool cannot be extended with customized elements without violating standards compliance (as is possible with some competing standards, such as CC/PP). However, this may also be an advantage in many scenarios: all elements are unambiguously described by the XML Schema and by normative text, so UED parsers know exactly what to expect, and how every element is to be interpreted.

The number of elements that can be described by UED is significant and many of them are rarely used. In practice, an application will almost always restrict itself to using a subset of elements and will add some additional restrictions. Other elements will be ignored and for missing elements some default value will be inferred. For example, when dealing with terminal descriptions, a terminal can have any number of displays; most application domains however will assume a single display (which can, by the way, still support multiple resolutions) and will not support UED documents which contain multiple Display sections. The SUIT project, too, will specify such a minimal subset to describe the terminal capabilities and network conditions. Additional elements can always be included by implementers should they have a need for it. A sample UED description for a SUIT terminal is given below:

```
<DIA xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
schemaLocation="urn:mpeg:mpeg21:2003:01-DIA-NS UsageEnvironment.xsd"
xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS">
 <Description xsi:type="UsageEnvironmentType">
  <UsageEnvironmentProperty xsi:type="TerminalsType">
   <Terminal>
    <TerminalCapability xsi:type="CodecCapabilitiesType">
     <Decoding xsi:type="VideoCapabilitiesType">
      <Format href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:4">
       <mpeg7:Name xml:lang="en" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001">
         MPEG-4
       </mpeg7:Name>
      </Format>
      <CodecParameter xsi:type="CodecParameterBitRateType">
       <BitRate average="5000000" maximum="20000000" />
      </CodecParameter>
     </Decoding>
```

```
    </TerminalCapability>
    <TerminalCapability xsi:type="DisplaysType">
     <Display>
      <DisplayCapability xsi:type="DisplayCapabilityType" bitsPerPixel="8"
                         colorCapable="true" activeDisplay="true">
       <Mode refreshRate="25">
        <Resolution horizontal="352" vertical="288" />
        <Resolution horizontal="176" vertical="144" />
       </Mode>
       <ScreenSize horizontal="80" vertical="50" />
       <RenderingFormat>Progressive</RenderingFormat>
      </DisplayCapability>
     </Display>
    </TerminalCapability>
    <TerminalCapability xsi:type="DeviceClassType">
     <DeviceClass href="urn:mpeg:mpeg21:2003:01-DIA-DeviceClassCS-NS:1">
      <mpeg7:Name xml:lang="en" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001">
        PDA
      </mpeg7:Name>
     </DeviceClass>
    </TerminalCapability>
    <TerminalCapability xsi:type="PowerCharacteristicsType"
     averageAmpereConsumption="2.5" batteryCapacityRemaining="5"
     batteryTimeRemaining="7200" runningOnBatteries="true" />
   </Terminal>
  </UsageEnvironmentProperty>
  <UsageEnvironmentProperty xsi:type="NetworksType">
   <Network>
    <NetworkCharacteristic xsi:type="NetworkCapabilityType"
      maxCapacity="20000000" minGuaranteed="64000" inSequenceDelivery="false"
      errorDelivery="false" errorCorrection="false" />
    <NetworkCharacteristic xsi:type="NetworkConditionType"
                           startTime="…" duration="…">
     <AvailableBandwidth average="8000000" />
     <Delay packetTwoWay="200" packetOneWay="100" delayVariation="10" />
     <Error packetLossRate="0.05" />
    </NetworkCharacteristic>
   </Network>
  </UsageEnvironmentProperty>
 </Description>
</DIA>
```

As can be seen, the description contains a terminal part as well as a networks part. User and natural environment characteristics are not deemed necessary for inclusion into the minimal subset. The terminal part consists of a CodecCapabilities, a Displays, a DeviceClass and a PowerCharacteristics subsection. The network part is subdivided into a NetworkCapability and a NetworkCondition subpart, describing the static and the dynamic behaviour of the network, respectively. This example UED document describes a terminal capable of receiving up to 20 Mbps video streams, having a color display of 8 bpp colour depth capable of 25 fps progressive and with resolutions of 352x288 and 176x144, a screen size of 80x50 mm, the device class is ‘PDA’, the terminal has 2h battery autonomy left, and is running on batteries. The network connection may vary between 64 kbps and 20 Mbps and the currently available bandwidth is an average 8 Mbps with a 200 ms round-trip delay, a delay jitter of 10 ms, and a packet loss rate of 5%.

4.2 Network Transport of MPEG-21 Descriptions

The MPEG-21 multimedia framework is highly suited for the description of the usage context. However, it is important to note that the MPEG-21 tools do not specify protocols, but rather file formats (in the form of XML Schemas) for these descriptions. In practice, these XML documents will need to be transported over a multitude of heterogeneous networks. By design, MPEG-21 does not specify how this must be done, since it is intended to be independent of network protocols and issues. In practice, IETF or W3C protocols are often used to transport MPEG-21 descriptions between nodes. This sometimes means that modifications or extensions to these protocols are necessary. For example, IETF SDPng can be extended to allow MPEG-21

UED fragments to be embedded inside [m12002]. As another example, MPEG-21 documents can be transported by means of RPC calls, using an interoperable RPC mechanism such as W3C's Simple Object Access Protocol (SOAP) over HTTP.

In the following sections, two mechanisms for transporting MPEG-21 documents are described in detail: using SOAP as an out-of-band channel; and extending RTSP to send the documents in-band with the RTSP session. Other means (which we won't discuss here) are possible, such as plain TCP socket programming or other RPC technologies. In the SUIT demonstrators, the SOAP approach will be used.

4.2.1 The Simple Object Access Protocol (SOAP)

A possible solution to transporting an MPEG-21 document from a terminal to a server is to make the terminal perform a remote procedure call (RPC) to the server, supplying the MPEG-21 document as an argument to the call. Any RPC mechanism capable of carrying XML documents as argument types is thus a candidate solution for this problem.

The Simple Object Access Protocol (SOAP) is an XML-based W3C protocol which meets these requirements. What's more, SOAP is widely used in many areas, such as Web Services. It is therefore well supported on many platforms and programming languages.

SOAP uses other standard L5 protocols for message passing, usually the Hypertext Transfer Protocol (HTTP); we then speak of a Web Service (WS). As a result SOAP, as the name suggests, is conceptually simpler than for example CORBA. It is also completely platform independent, and it is firewall-friendly, since most firewalls allow HTTP connections to be made.

The following transcript illustrates a simple SOAP Remote Procedure Call to a function with `String register();` prototype:

```
Client:
POST /axis/services/Playout HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
2.0.50727.42)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Host: server:8080
Content-Length: 542
Expect: 100-continue
Connection: Keep-Alive



Server:
HTTP/1.1 100 Continue

Client:
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
               mlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
               xmlns:tns="http://localhost:8080/axis/services/Playout"
       xmlns:types="http://localhost:8080/axis/services/Playout/encodedTypes"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <q1:register xmlns:q1="http://mmlab.ugent.be" />
  </soap:Body>
</soap:Envelope>

Server:
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/xml;charset=utf-8
Transfer-Encoding: chunked
Date: Wed, 27 Sep 2006 09:57:13 GMT
```

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:registerResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://mmlab.ugent.be">
      <registerReturn xsi:type="xsd:string">
        13af08901d66093b1387e738e4258ff11c
      </registerReturn>
    </ns1:registerResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

In this example, the client initiates a HTTP connection and POSTs a SOAP RPC request inside the HTTP body, which is of mime-type `text/xml`. The SOAP request is an XML document containing the remote call's name ("register") and its arguments, if any. The server's reply is another SOAP XML document, containing the return value of the remote call (in this case, a string containing an identifier).

4.2.1.1 Sending XML Documents using SOAP

Arguments to SOAP Remote Procedure Calls can also consist of entire XML documents, formatted as a single (very long) string. Alternatively, some SOAP implementations support the relatively new "SOAP with attachments" mechanism, by which the argument in question is sent as a MIME attachment. The HTTP body as a whole is then of mime-type `multipart/related` instead of simply `text/xml`. The following SOAP request illustrates the first technique, where the argument is inlined as a very long string. As can be seen, the XML reserved tokens, mainly `<` and `>`, are escaped using HTML-style escape sequences. This request transcript calls the remote function with this prototype: `int update(int sessionId, String diaDocument);`

```
POST /axis/services/Playout HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; MS Web Services Client Protocol
2.0.50727.42)
Content-Type: text/xml; charset=utf-8
SOAPAction: ""
Host: hindemith:8080
Content-Length: 4196
Expect: 100-continue


<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
               xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
               xmlns:tns="http://localhost:8080/axis/services/Playout"

xmlns:types="http://localhost:8080/axis/services/Playout/encodedTypes"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <q1:update xmlns:q1="http://mmlab.ugent.be">
      <sessionId xsi:type="xsd:string">
        130890156609301387873874258281115
      </sessionId>
      <diaDocument xsi:type="xsd:string">
        &lt;?xml version="1.0" encoding="utf-16" standalone="no"?&gt;
        &lt;DIDL xmlns="urn:mpeg:mpeg21:2002:02-DIDL-NS"&gt;
          &lt;Item&gt;
            &lt;Descriptor&gt;
              &lt;Statement mimeType="text/xml"&gt;
                &lt;DIA xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                        schemaLocation="urn:mpeg:mpeg21:2003:01-DIA-NS
UsageEnvironment.xsd"
                        xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS"&gt;
```

```
                  &lt;Description xsi:type="UsageEnvironmentType"&gt;
                    &lt;UsageEnvironmentProperty xsi:type="TerminalsType"&gt;
                      &lt;Terminal&gt;
                        &lt;TerminalCapability
xsi:type="CodecCapabilitiesType"&gt;
                          &lt;Decoding xsi:type="VideoCapabilitiesType"&gt;
                            &lt;Format
href="urn:mpeg:mpeg7:cs:VisualCodingFormatCS:2001:4"&gt;
                              &lt;mpeg7:Name xml:lang="en"

xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"&gt;MPEG-4 SVC
                              &lt;/mpeg7:Name&gt;
                            &lt;/Format&gt;
                            &lt;CodecParameter
xsi:type="CodecParameterBitRateType"&gt;
                              &lt;BitRate  average="5000000"  maximum="10000000"
/&gt;
                            &lt;/CodecParameter&gt;
                          &lt;/Decoding&gt;
                        &lt;/TerminalCapability&gt;
                        &lt;TerminalCapability xsi:type="DisplaysType"&gt;
                          &lt;Display&gt;
                            &lt;DisplayCapability
xsi:type="DisplayCapabilityType"
                                                bitsPerPixel="8"
                                                colorCapable="true"
                                                activeDisplay="true"&gt;
                              &lt;Mode refreshRate="25"&gt;
                                &lt;Resolution  horizontal="720"  vertical="576"
/&gt;
                                &lt;Resolution  horizontal="352"  vertical="288"
/&gt;
                              &lt;/Mode&gt;
                              &lt;ScreenSize   horizontal="40"   vertical="30"
/&gt;

&lt;RenderingFormat&gt;Progressive&lt;/RenderingFormat&gt;
                            &lt;/DisplayCapability&gt;
                          &lt;/Display&gt;
                        &lt;/TerminalCapability&gt;
                        &lt;TerminalCapability xsi:type="DeviceClassType"&gt;
                          &lt;DeviceClass    href="urn:mpeg:mpeg21:2003:01-DIA-
DeviceClassCS-NS:1"&gt;
                            &lt;mpeg7:Name xml:lang="en"

xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"&gt;PDA
                            &lt;/mpeg7:Name&gt;
                          &lt;/DeviceClass&gt;
                        &lt;/TerminalCapability&gt;
                        &lt;TerminalCapability
xsi:type="PowerCharacteristicsType"
                                                averageAmpereConsumption="2.5"
                                                batteryCapacityRemaining="5"
                                                batteryTimeRemaining="7200"
                                                runningOnBatteries="true" /&gt;
                      &lt;/Terminal&gt;
                    &lt;/UsageEnvironmentProperty&gt;
                    &lt;UsageEnvironmentProperty xsi:type="NetworksType"&gt;
                      &lt;Network&gt;
                        &lt;NetworkCharacteristic
xsi:type="NetworkCapabilityType"
                                                maxCapacity="512000"
                                                minGuaranteed="0"
                                                inSequenceDelivery="false"
                                                errorDelivery="false"
                                                errorCorrection="false"
/&gt;
                        &lt;NetworkCharacteristic
xsi:type="NetworkConditionType" startTime="" duration=""&gt;
                          &lt;AvailableBandwidth average="0" /&gt;
                          &lt;Delay    packetTwoWay="200"    packetOneWay="100"
```

```
delayVariation="10" /&gt;
                          &lt;Error packetLossRate="0.05" /&gt;
                        &lt;/NetworkCharacteristic&gt;
                      &lt;/Network&gt;
                    &lt;/UsageEnvironmentProperty&gt;
                  &lt;/Description&gt;
                &lt;/DIA&gt;
              &lt;/Statement&gt;
            &lt;/Descriptor&gt;
          &lt;/Item&gt;
        &lt;/DIDL&gt;
      </diaDocument>
    </q1:update>
  </soap:Body>
</soap:Envelope>
```

4.2.2 Extending RTSP for Transportation of Usage Environment Descriptions

[Informative] In unicast scenarios, an active RTSP session between a terminal and the playout is already present. Therefore it makes sense to extend the Real Time Streaming Protocol (RTSP) so that UED documents can be transmitted in-band inside the RTSP session. RTSP can be extended in three ways. Section 1.5 of the RFC states:

> RTSP can be extended in three ways, listed here in order of the magnitude of changes supported:
>
> - Existing methods can be extended with new parameters, as long as these parameters can be safely ignored by the recipient. […]
> - **New methods can be added.** If the recipient of the message does not understand the request, it responds with error code 501 (Not implemented) and the sender should not attempt to use this method again. A client may also use the OPTIONS method to inquire about methods supported by the server. The server SHOULD list the methods it supports using the Public response header.
> - A new version of the protocol can be defined […]

We propose to use the option in bold. Therefore, a new ENVIRONMENT method is proposed, illustrated by the following sample:

Request: Client->Server

```
ENVIRONMENT * RTSP/1.0
CSeq: 343
Session: 13759028
Content-Type: text/xml
Content-Length: XXXX

<DIDL>
 <!-- ...the UED document... -->
</DIDL>
```

Response: Server->Client

```
RTSP/1.0 200 OK
CSeq: 343
Date: 24 Nov 2006 10:23:34 GMT
```

The RTSP server will, upon reception, forward this UED document to the AdaptationQoS engine which in turn drives the stream extractor. Restrictions are that every terminal participating in a unicast streaming session must send an ENVIRONMENT message before any PLAY command is sent. Terminals can re-send their UEDs (identical or updated) at any time they wish to, as long as their RTSP session is still valid.

4.3 QoS Adaptation through SVC Bitstream Extraction

Once feedback information has been relayed to the playout and the UED document has been parsed, the obtained information can be used to control the bitstream extractor process. Since the extractor itself contains no intelligence (it just removes temporal and spatial layers, and truncates FGS data), the feedback information must be converted into parameters for steering the extractor. MPEG-21 AdaptationQoS, described below, offers a flexible solution for performing this conversion.

4.3.1 MPEG-21 DIA AdaptationQoS

In MPEG-21 DIA, terminal and network quality of service (QoS) addresses the problem of selecting optimal parameter settings for media resource adaptation to satisfy constraints imposed by terminals and/or networks while maximizing the quality of service. AdaptationQoS is an MPEG-21 tool to specify the relationship between constraints and the feasible adaptation operations to satisfy these constraints. As such, this tool provides an interoperable means to specify how a media resource can be adapted, given certain constraints. For example, given constraints such as currently available bandwidth and maximum terminal display resolution, we can use AdaptationQoS to describe how many spatial, temporal, and/or quality layers should be removed from a scalable bitstream in order to satisfy these constraints. The XML document below represents an AdaptationQoS description for such a scenario:

```
        <DIA xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             schemaLocation="urn:mpeg:mpeg21:2003:01-DIA-NS
UsageEnvironment.xsd"
             xmlns="urn:mpeg:mpeg21:2003:01-DIA-NS">
          <DescriptionMetadata>
            <ClassificationSchemeAlias alias="AQoS"
              href="urn:mpeg:mpeg21:2003:01-DIA-AdaptationQoSCS-NS"/>
            </DescriptionMetadata>
          <Description xsi:type="AdaptationQoSType">
            <Module xsi:type="UtilityFunctionType">
              <Constraint iOPinRef="BANDWIDTH">
                <Values xsi:type="IntegerVectorType">
                  <Vector>100 200 400 800 1600 3200 6400 12800</Vector>
                </Values>
              </Constraint>
              <AdaptationOperator iOPinRef="SPATIALLAYERS">
                <Values xsi:type="IntegerVectorType">
                  <Vector>2 2 1 1 0 0 0 0</Vector>
                </Values>
              </AdaptationOperator>
              <AdaptationOperator iOPinRef="TEMPORALLEVELS">
                <Values xsi:type="IntegerVectorType">
                  <Vector>3 2 3 2 2 1 0 0</Vector>
                </Values>
              </AdaptationOperator>
              <AdaptationOperator iOPinRef="QUALITYREDUCTION">
                <Values xsi:type="FloatVectorType">
                  <Vector>1.00 1.00 0.80 0.60 0.40 0.20 0.50 0</Vector>
                </Values>
              </AdaptationOperator>
              <UtilityRank>1 2 3 4 5 6 7 8</UtilityRank>
            </Module>
            <IOPin semantics="AQoS:6.6.5.1" id="BANDWIDTH"/>
            <IOPin semantics="AQoS:6.5.9.1" id="HORIZONTALSCREENSIZE"/>
            <IOPin semantics="AQoS:6.5.9.2" id="VERTICALSCREENSIZE"/>
            <IOPin semantics="AQoS:1.3.9.1" id="SPATIALLAYERS"/>
            <IOPin semantics="AQoS:1.3.9.2" id="TEMPORALLEVELS"/>
            <IOPin semantics="AQoS:1.3.9.3" id="QUALITYREDUCTION"/>
          </Description>
        </DIA>
```

As can be seen from this example, an AdaptationQoS description consists of one or more Modules plus a set of IOPins. A Module specifies a mapping between Constraints (which are the input parameters to the

module) and AdaptationOperators (which are its output parameters). IOPins are used to identify these inputs and outputs, thereby enabling the interconnection of one Module's outputs to another Module's inputs. Hence, in the most general case, many Modules can be chained together in order to realize very complex mappings.

Several types of Modules are defined: the LookUpTable implements a mapping by way of a look-up matrix with several axes; the StackFunction places the values of the input Constraints on a stack and can perform arithmetic on the topmost elements of the stack, implementing a kind of Polish notation calculator; and the UtilityFunction (used in this example) assigns AdaptationOperator values based on a list of user-supplied key points with associated utility (subjective or objective quality measure).

In this simple example, only one Module is used, taking the BANDWIDTH IOPin as its input Constraint. The IOPin declaration of the BANDWIDTH IOPin contains a reference to the semantics of the IOPin; these are taken from the AdaptationQoSCS Classification Scheme which is a normative part of MPEG-21 DIA and which unambiguously identifies the IOPin as representing the minimum available bandwidth taken from the network condition part of the UED:

```
<Term termID="6.6.5.1">
    <Name xml:lang="en">AvailableBandwidth minimum</Name>
    <Definition xml:lang="en">Describes the minimum AvailableBandwidth as
defined in this part of ISO/IEC 21000.</Definition>
</Term>
```

The three output IOPins of the Module represent the number of spatial layers, the number of temporal layers, and the amount of quality information that must be discarded from the bitstream in order to meet the imposed bandwidth constraint. An adaptation engine that implements the UtilityFunction Module will compute these outputs based on the input constraint and on the Utility or UtilityRank values supplied in the description, typically by finding the point with the highest Utility or UtilityRank satisfying all constraints. It is also possible for an adaptation engine to interpolate between constraint points, at least for the AdaptationOperators for which this makes sense.

4.3.2 Implementation of AdaptationQoS Engines

As mentioned before, actual adaptation engines are not within the scope of MPEG-21 standardization. Instead, an AdaptationQoS-based adaptation engine will retrieve and parse the AdaptationQoS document that applies to the media being adapted. Based on this, the engine will configure itself; then it can commence adaptation of the media, resulting in an adapted Digital Item. Fig. 4.3 illustrates this work flow.

The AdaptationQoS description could come from different sources, depending upon the application. It could be inlined inside the DID of the Digital Item, so every Digital Item would have its private AdaptationQoS description. Alternatively, the DID could contain an URI to the AdaptationQoS document that is to be used; the document itself could then be stored in a network-accessible location such as a web server. For example, this way, one AdaptationQoS document per genre could be used, each of which could be tailored to best suit the needs of that particular genre. (E.g., action movies might sacrifice spatial layers rather than temporal layers, while the inverse might be true for documentaries).

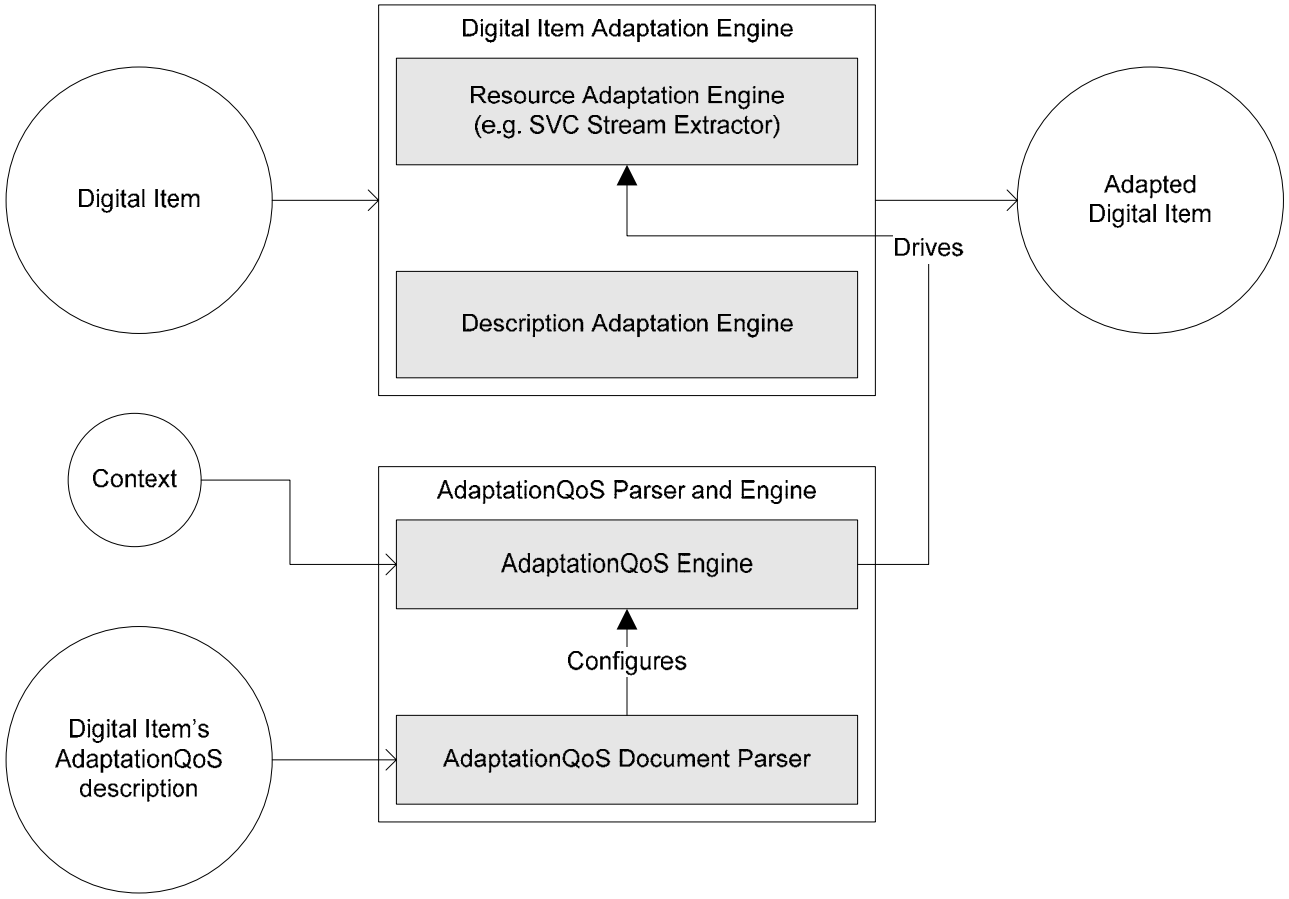


Figure 4. Workflow in an AdaptationQoS Implementation

The input marked 'Context' in the figure denotes the sources of the input IOPins (the Constraints) used in the AdaptationQoS description. If flexibility is important, BSDLink [iso21000] can be used to allow the engine to source the necessary inputs at run time; else, if the inputs are a small set of known entities (such as in SUIT, the available bandwidth plus the horizontal and vertical screen size from the terminal's UED) they can just be 'hardwired' inside the engine implementation.

4.3.3 Defining the Adaptation Operators Needed By the Bitstream Extractor

The general idea when using MPEG-21 AdaptationQoS is to maintain a clean separation between policy and mechanism, i.e. to keep the actual bitstream extractor as 'dumb' as possible, and to encode all intelligence inside the AdaptationQoS XML document. The AdaptationQoS document is therefore sometimes referred to as the *steering description* of the bitstream adaptation engine.

Maintaining this separation implies that a set of output AdaptationOperators will have to be chosen in such a way that the bitstream extractor can be relatively simply, and such that there is no decision-making required inside the extractor. Since the easiest extraction mechanism consists of the removal of entire layers, we start by defining two AdaptationOperators, *temporalLevels* and *spatialLayers*, indicating the number of temporal and spatial layers that need to be removed from the bitstream.

The remainder of the bitstream adaptation process now consists of removing part of the FGS information, by truncation of FGS NAL units, in order to reach the targeted bit rate. This process is driven by a third AdaptationOperator we call *qualityReduction*. It indicates the amount of FGS data, in bps, that must be removed in order to produce the final adapted bitstream.

4.3.4 AdaptationQoS Proposal for SUIT

In the SUIT project, we propose the following AdaptationQoS implementation. As discussed above, the bitstream extractor is driven by three parameters: *temporalLevels* and *spatialLayers*, indicating the number of entire layers to be removed; and *qualityReduction*, indicating the amount of FGS information to truncate. In unicast scenarios, the input parameters to the AdaptationQoS engine are extracted from the Usage Environment parameters that are supplied by the terminal; these are:

- The average available bandwidth;
- The maximum horizontal display resolution;
- The maximum vertical display resolution.

In broadcast scenarios, a single parameter will be used: the target bitrate. In this case, this parameter is generated by the playout, which estimates the bitrate to be allocated to a stream based on the amount of streams to be broadcast and the amount of other information (internet, hyperlinked video, …) that has to be sent alongside the regular broadcasts.

A single UtilityFunction module will be sufficient to map these input parameters to the three adaptation operators that drive the bitstream extractor. This UtilityFunction module will be set up so that:

1. (Unicast only:) For defining *spatialLayers*, it gives precedence to the maximum display resolution. This way, the adapted bitstream's resolution is never greater than the maximum resolution the terminal can display;
2. Then temporal and/or more spatial layers are removed based on the available bandwidth. Which layers are to be removed first is expressed in the AdaptationQos file.
3. Finally, *qualityReduction* is calculated in order to fill the remainder of the available bandwidth.

For the SUIT demonstrator, the MPEG-21 parser and AdaptationQoS engine will be integrated in a Java web service, using a SOAP RPC front end. In the unicast scenarios, terminals can periodically submit their UED document by invoking a SOAP remote procedure call on the web service. In the broadcast scenario, an RPC method will be provided through which other parts of the gateway can set the target bitrate. Inside the playout, an inter-process communication (IPC) channel is required between the web service and the bitstream extractors it has under its control. For this purpose, a simple UDP or TCP based message system will be employed; compared to other IPC mechanisms, this has the advantage that web service and extractor do not even have to run on the same machine, resulting in a playout system that scales better.

5 Protocols in SUIT

5.1 Introduction

In scenarios with a SUIT gateway, session protocols need to be defined for two distinct transmissions of video:

1. Between playout and gateway (last-mile networks, DVB & WiMAX)
2. Between gateway and terminals (home network, WLAN)

The playout does not differentiate between gateway and no-gateway scenarios. Therefore, in scenarios where no gateway is present the terminal will include the necessary gateway functionality in order to communicate directly with the playout. In these scenarios, we must therefore only consider (1) above.

5.2 Communication between Playout and Gateway (Last-Mile Networks)

5.2.1 Session Description

For session description (which includes service names, location, streaming attributes, coding format details, description of layers, …) the following candidate protocols have been selected and described:

- Session Description Protocol (SDP)
- MPEG-21 DID
- SD&S records
- SDP new generation (SDPng)

In SUIT, SDP will be used for session announcements, based on the following merits:

- There is good software support (in contrast to e.g. SDPng);
- An extension to SDP exists facilitating the description of layers in a scalable bitstream or descriptions in Multiple Description Coded (MDC) video. The extension allows an explicit syntax for describing the dependencies between layers/descriptions;
- It is tailored for use in an all-IP streaming media (RTP) environment, as opposed to e.g. the generic MPEG-21 DID.

5.2.2 Session Announcement

To announce the session description data to the terminals, two solutions were put forth:

- Session Announcement Protocol (SAP)
- SD&S announcement

SD&S is capable of announcing not only video but also other content such as MHP applications. In order to have a single solution for the entire project, it was decided to use the announcement mechanisms of SD&S in all scenarios. Since the SD&S standard uses its own payload format for session description, the protocol will have to be extended so that the payload can consist of a pointer (e.g. an URI) to the actual session descriptions in SDP files. The SDP files themselves will be made accessible to the terminals for HTTP retrieval or will be sent in a RTSP session.

In order to minimise start-up delay of the terminals, session announcement will proceed like this:

1. Terminal discovers playout using the SD&S discovery mechanism (see e.g. D1.1)
2. Terminal gets a list of available programmes (which contains no detailed information)

3. This shortlist allows the user to selects the desired programme
4. The detailed SDP file will then be retrieved and the RTP streaming session can be initiated.

5.2.3 Feedback Channel

This subsection applies to unicast scenarios. To convey return channel information, such as network conditions and terminal capabilities, back to the playout, the following protocols were discussed:

- Tools from MPEG-21 DIA
- Real Time Control Protocol (RTCP)

RTCP has some large disadvantages over MPEG-21 DIA:

- Within the SUIT context, it focuses on network condition too much. To describe terminal capabilities such as screen size or battery life, RTCP should have to be extended; MPEG-21 DIA tools can describe all this (and much more) out-of-the-box.
- RTCP is tied to RTP as a 'companion protocol'. As such, its use would impose a stricter-than-necessary binding on some modules of the playout (e.g. extractor and RTP encapsulator). MPEG-21 DIA, in contrast, offers a higher-level, general framework which is independent of the actual transport mechanism.

Therefore it was decided to use the MPEG-21 DIA framework in the return path. MPEG-21 Usage Environment Description (UED) documents will be generated by the gateway/terminal in order to describe the feedback information. This information will then periodically be sent to the playout. MPEG-21 AdaptationQoS will be used in the playout to drive the bitstream extractor; the specifics are described in Section 4.3.4.

5.3 Communication between Gateway and Terminal (Home Network)

For the streaming of video between a gateway and the terminals it serves, simplicity is advised. SDP is proposed as the description protocol. Since the video output by the gateway is single description SVC, the SDP files used here will be different from those issued by the playout. The gateway will thus be responsible for generating minimalist SDP files, containing at least the RTP parameters and port number, so the terminal can access the video stream.

The gateway can perform additional bitstream adaptation to adjust bitrate to the conditions of the local WLAN. In order not to complicate matters, it is advised to use RTCP as the feedback mechanism here instead of MPEG-21 DIA. Since there are existing streaming libraries containing RTCP functionality, this will allow for a rapid implementation of the streaming connectivity between gateway and terminal. A single 3rd party library can then be used to perform all RTP, RTSP and RTCP related work.

5.4 Sequence Diagram of Terminal Start-Up

The sequence diagram below illustrates the steps, relevant to A4.2, that are taken by a gateway after it is powered on. First, the gateway executes the SD&S service discovery steps in order to locate the SUIT playout services. It will then have acquired a list of available TV programmes. This list contains no detailed information, so it is relatively small, which helps in minimising the power-up delay. If a terminal on the WLAN wishes to receive a certain programme, the gateway will acquire the detailed session description from the playout using SD&S. This detailed session description is an SDP file.

The information in the SDP file allows the gateway to initiate a streaming session. At the same time the gateway will have to inform the terminal about the RTP parameters on the last hop (i.e. the local WLAN). For the last hop, this task can be as much as possible offloaded to existing solutions, e.g. RTP/SDP/RTSP/RTCP libraries. This will likely result in a small SDP description sent to the terminal.

Upon reception of both descriptions, the gateway can start combining NAL units. The resulting single description video stream can then be broadcast on the local WLAN. The terminal receives it simply by listening to the appropriate broadcast stream. Concurrently the gateway starts maintaining an UED document and periodically invokes the AdaptationQoS web service in order to inform the gateway of the condition of the last-mile networks.

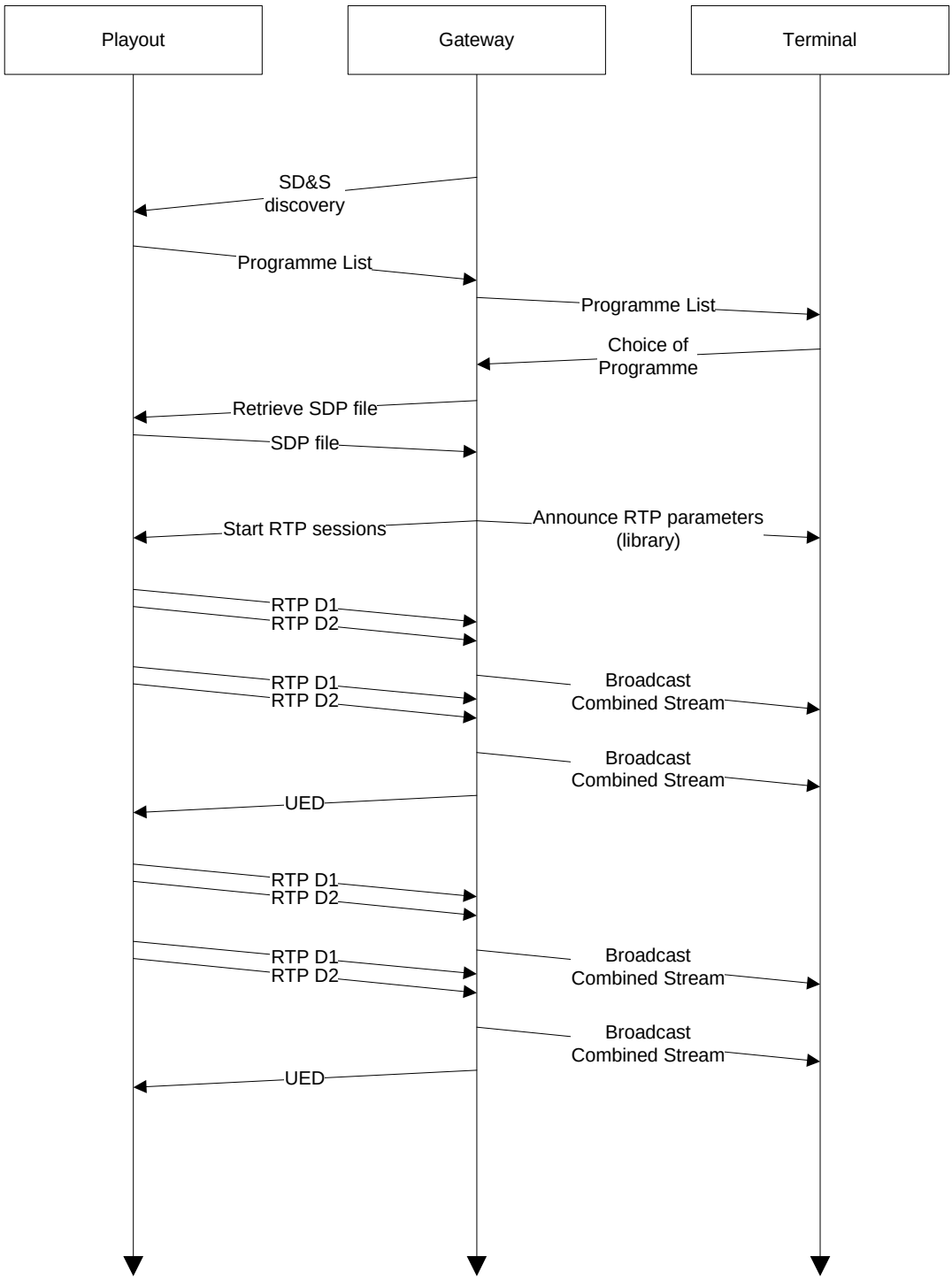


Figure 5. Sequence Diagram of Gateway and Terminal Power-Up

6 Conclusions

In order to set up a complete delivery chain for digital video, protocols and document formats have to be chosen for session description, announcement and managing, and usage environment feedback. For each of these tasks, several candidate solutions have been presented:

- Session description formats:
  - The Session Description Protocol (SDP);
  - The Session Description Protocol, new generation (SDPng);
  - The MPEG-21 Digital Item Declaration Language (DIDL);
  - The Service Discovery & Selection protocol (SD&S).
- Session announcement and management protocols:
  - The Session Announcement Protocol (SAP);
  - The Internet Group Management Protocol (IGMP);
  - The Real Time Streaming Protocol (RTSP);
  - The Service Discovery & Selection protocol (SD&S).
- Usage Environment Description formats and feedback transport protocols:
  - MPEG-21 DIA Usage Environment Description (DIA-UED);
  - The Simple Object Access Protocol (SOAP);
  - QoS functionalities of the Real Time Control Protocol (RTCP).

Among these proposed solutions, the most appropriate have been identified, based on the project's requirements and the expertise of the Consortium partners. For the communication between the playout system and gateways, which is the main focus of this deliverable, the following technologies were chosen:

- The session description format will be SDP. It can be easily parsed, is well suited to convey network-related parameters (such as RTP attributes), and has explicit support for signaling MDC relationships between streams;
- Session announcement will be dealt with by SD&S. The protocol defines its own session description format, so it will be modified so as to allow interoperation with SDP. Instead of its own description format, the announcements will consist of a pointer to the relevant SDP description file, which is network-accessible to the gateways, e.g. by HTTP retrieval. This results in a two-step announcement procedure, where first a shortlist of available programmes is announced to all clients, and later the detailed descriptions are accessed on an as-needed basis.
- MPEG-21 DIA-UED will be used to describe the usage environment. RTCP's vocabulary is too limited, and it is tied to the lower-level protocols too much. MPEG-21 offers a high-level solution and is decoupled from the forward transmission protocols. Besides DIA-UED, DIA-AdaptationQoS will be employed to steer the bitstream extraction process at the playout side.

Concerning the communication between a gateway and terminals, i.e. on the home WLAN, a simpler solution is advised. RTP is proposed, in combination with simple SDP files conveying just enough information to allow terminals to receive the video streams. In cases where the gateway performs additional rate adaptation, the QoS provisions of RTCP can be used to inform the gateway of the WLAN conditions. The objective is to allow, as much as possible, the streaming on the home WLAN to be based on existing integrated RTP/RTSP/RTCP solutions.

7 Acronyms

AQoS	AdaptationQoS
CORBA	Common Object Request Broker Architecture
DI	Digital Item
DIA	Digital Item Adaptation
DID	Digital Item Declaration
DIDL	Digital Item Declaration Language
DVB	Digital Video Broadcast
DVBSTP	Digital Video Broadcast – SD&S Transport Protocol
ETSI	European Telecommunications Standards Institute
FGS	Fine-Grain Scalability
FTP	File Transfer Protocol
HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
IEC	International Engineering Consortium
IETF	Internet Engineering Task Force
IGMP	Internet Group Management Protocol
ISO	International Organization for Standardization
MDC	Multiple Description Coding
MIME	Multipurpose Internet Mail Extensions
MPEG	Moving Picture Experts Group
NAL	Network Abstraction Layer
NALU	Network Abstraction Layer Unit
QoS	Quality of Service
RFC	Request For Comments
RTCP	Real Time Control Protocol
RTP	Real Time Protocol
RTSP	Real Time Streaming Protocol
SAP	Session Announcement Protocol
SD&S	Service Discovery and Selection
SDP	Session Description Protocol
SDPng	Session Description Protocol, new generation
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SVC	Scalable Video Coding
UED	Usage Environment Description
W3C	World Wide Web Consortium
WLAN	Wireless Local Area Network
WS	Web Service
XML	Extensible Mark-up Language

8 References

[rfc2326] Schulzrinne, H., Rao, A., and Lanphier, R., 1998. Real Time Streaming Protocol (RTSP). IETF, Request for Comments, RFC 2326.

[rfc2974] Handley, M., Perkins, C., and Whelan, E., 2000. Session Announcement Protocol. IETF, Request for Comments, RFC 2974.

[ts102034] 2006. Digital Video Broadcasting (DVB); Transport of MPEG-2 Based DVB Services over IP Based Networks. ETSI TS 102 034 v1.2.1.

[rfc2236] Fenner, W., 1997. Internet Group Management Protocol. IETF, Request for Comments, RFC 2236.

[rfc3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and Thyagarajan, A., 2002. Internet Group Management Protocol, Version 3. IETF, Request for Comments, RFC 3376.

[rfc2365] Mayer, D., 1998. Administratively scoped IP multicast. IETF, Request for Comments, RFC 2365.

[rfc3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E., "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[rfc4566] Handley, M., Jacobson, V., and Perkins, C., Jul. 2006. SDP: Session Description Protocol, IETF, Request for Comments, RFC 4566.

[rfc3984] Wenger, S., Hannuksela, M.M., Stockhammer, T., Westerlund, M., and Singer, D., Feb. 2005. RTP Payload Format for H.264 Video. IETF, Request for Comments, RFC 3984.

[rfc3550] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V., Jul. 2003. RTP: A Transport Protocol for Real-Time Applications. IETF, Request for Comments, RFC 3550.

[rfc3551] Schulzrinne, H. and Casner, S., Jul. 2003. RTP Profile for Audio and Video Conferences with Minimal Control. IETF, Request for Comments, RFC 3551.

[rfc3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and Norrman, K., Mar. 2004. The Secure Real-time Transport Protocol (SRTP). IETF, Request for Comments, RFC 3711.

[rfc1305] Mills, D., Mar. 1992. Network Time Protocol (Version 3) Specification, Implementation. IETF, Request for Comments, RFC 1305.

[rfc3388] Camarillo, G., Holler, J., and H. Schulzrinne, Dec. 2002. Grouping of Media Lines in the Session Description Protocol (SDP). IETF, Request for Comments, RFC 3388.

[iso21000] Moving Picture Experts Group Mar. 2003. Information technology — Multimedia framework (MPEG-21) — Part 7: Digital Item Adaptation. ISO/IEC JTC1/SC29/WG11 FDIS 21000-7.

[jsvm4] Joint Video Team, Jul. 2006. Joint Scalable Video Model JSVM-4 Annex G. (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6)

[wenger] Wenger, S., Wang, Y.-K., Schierl, T., Oct. 2006. RTP Payload Format for SVC Video. IETF, Internet Draft, draft-wenger-avt-rtp-svc-03.

[bormann] Bormann, K.O., Feb. 2005. Session Description and Capability Negotiation. IETF, Internet Draft, draft-ietf-mmusic-sdpng-08.

[schierl] T. Schierl, T., December. 2006. Signaling of layered and multi description media in Session Description Protocol (SDP). IETF, Internet Draft, draft-schierl-mmusic-layered-codec-02.

[m10996] Wolf, I., Feiten, B., Guenkova-Luy, T., Schorr, A., Hauck, F., and Kassler, A.J., 2004. MPEG-21 DIA based delivery using SDPng and RTP. ISO/IEC JTC1/SC29/WG11/M10996.

[m12002] Guenkova-Luy, T., Schorr, A., Hauck, F., Kassler, A.J., Wolf, I., Feiten, B., Timmerer, C., and Hellwagner, H., 2005. MPEG-21 DIA based content delivery using SDPng controls and RTP transport. ISO/IEC JTC1/SC29/WG11 MPEG2004/M12002.