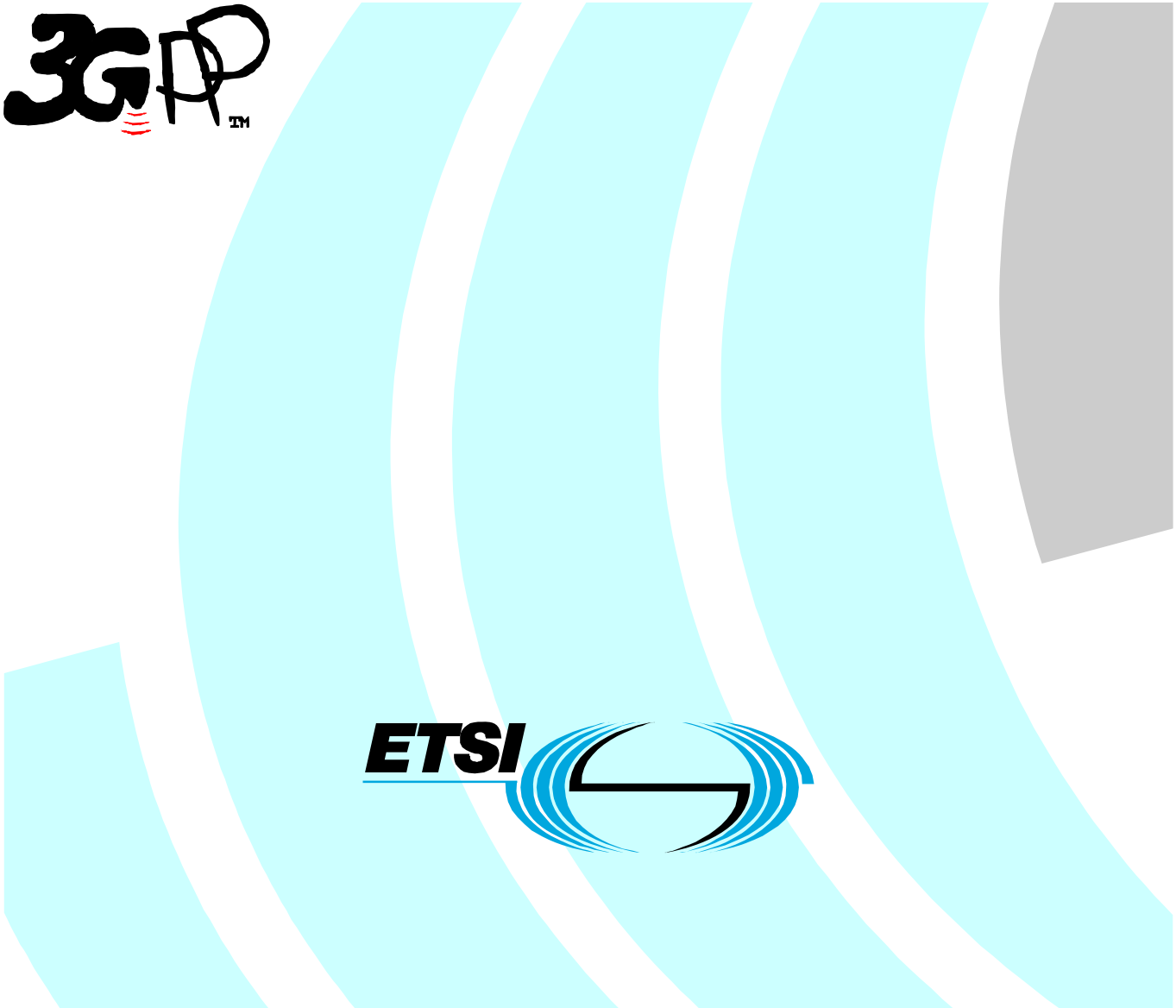
ETSI TS 126 142 V7.3.0 (2009-06)

Technical Specification

Universal Mobile Telecommunications System (UMTS);
LTE;
Dynamic and Interactive Multimedia Scenes (DIMS)
(3GPP TS 26.142 version 7.3.0 Release 7)



Reference
RTS/TSGS-0426142v730

Keywords
UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Specification (TS) has been produced by ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under http://webapp.etsi.org/key/queryform.asp.

Contents

Foreword

This Technical Specification has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

- x the first digit:
  - 1 presented to TSG for information;
  - 2 presented to TSG for approval;
  - 3 or greater indicates TSG approved document under change control.
- y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.
- z the third digit is incremented when editorial only changes have been incorporated in the document.

Introduction

Dynamic and Interactive Multimedia Scenes (DIMS) is a dynamic, interactive, scene-based media system which enables display and interactive control of multimedia data such as audio, video, graphics, images and text. It ranges from a movie enriched with vector graphic overlays and interactivity (possibly enhanced with closed captions), to complex multi-step services with fluid interaction/interactivity and different media types at each step. The demand for such Rich Media service is increasing at a high pace, spurred by the development of the next generation mobile infrastructure and the generalization of TV content to new mobile environments.

In the case of a video portal application, subscribers can watch TV, video and audio enriched with additional data (graphics, text, images) in streaming, progressive download or offline mode. DIMS provides a convenient and natural way to browse rich-media services, a web-like access (content available in less than three clicks, easy discovery, no learning curve), a permanent refresh of content through dynamic updates available on the fly and decreasing latency by allowing the visualization of data as soon as possible.

Content can be synchronized up to a frame-accurate basis (e.g. to ensure content providers and operators that voting will start and stop at a precise time during a vote within an interactive show or to allow karaoke text flows).

1 Scope

DIMS defines a dynamic rich-media system, including a media type, its packaging, delivery, and interaction with the local terminal, user, and other local and remote sub-systems. Enhanced end-user experiences are provided by the coordinated management and synchronization of media and events, combined with end-user interaction.

The DIMS media type can be used as a generic media type, allowing creating dynamic interactive rich-media services and can also benefit, or be used in association with other media types (e.g.: audio codecs, video codecs, XHTML browser, etc.).

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

[1] W3C Candidate Recommendation: "Scalable Vector Graphics (SVG) Tiny 1.2 Specification".

NOTE: Available at: http://www.w3.org/TR/SVGMobile12/.

[2] Open Mobile Alliance (July 2004): "ECMAScript Mobile Profile 1.0".

[3] ISO/IEC 14496-20:2006: "Information technology - Coding of audio-visual objects - Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)", including ISO/IEC 14496-20:2006/COR1, ISO/IEC 14496-20:2006/AMD1.

[4] ISO/IEC 14496-22: "Information technology - Coding of audio-visual objects - Part 22: Open Font Format".

[5] W3C Recommendation (December 2005): "Synchronized Multimedia Integration Language (SMIL 2.1)".

NOTE: Available at: http://www.w3.org/TR/2005/REC-SMIL2-20051213/.

[6] 3GPP TS 26.140: "Multimedia Messaging Service (MMS); Media format and codecs".

[7] 3GPP TS 26.234: "Transparent end-to-end Packet-switched Streaming Service (PSS); Protocols and codecs".

[8] 3GPP TS 26.244: "Transparent end-to-end packet switched streaming service (PSS); 3GPP file format (3GP)".

[9] The Unicode Consortium: "The Unicode Standard", Version 5.0, http://www.unicode.org/.

[10] ISO/IEC 14496-12: "Information technology - Coding of audio-visual objects - Part 12: ISO base media file format".

[11] IETF RFC 1952 (May 1996): "GZIP file format specification version 4.3", P. Deutsch.

[12] IETF RFC 2616 (June 1999): "Hypertext Transfer Protocol -- HTTP/1.1", R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee.

[13] IETF RFC 4329 (April 2006): "Scripting Media Types", B. Hoehrmann.

[14] IETF RFC 4281 (November 2005): "The Codecs Parameter for "Bucket" Media Types", R. Gellens, D. Singer, P. Frodjh.

[15] IETF STD 0064/RFC 3550 (July 2003) "RTP: A Transport Protocol for Real-Time Applications", H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson.

[16] IETF RFC 2326 (April 1998): "Real Time Streaming Protocol (RTSP)", H. Schulzrinne, A. Rao, R. Lanphier.

[17] W3C Document Object Model (DOM) Level 3 Events Specification, Version 1.0, W3C Working Draft 13 April 2006.

NOTE: Available at: http://www.w3.org/TR/DOM-Level-3-Events/.

[18] IETF RFC 2046 (November 1996): "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", N. Freed, N. Borenstein.

[19] W3C XML Events, an Events Syntax for XML, W3C Recommendation 14 October 2003.

NOTE: Available at: http://www.w3.org/TR/2003/REC-xml-events-20031014.

[20] W3C Media Access Events http://www.w3.org/TR/MediaAccessEvents/

[21] IETF STD 65, RFC 3551 (July 2003): "RTP Profile for Audio and Video Conferences with Minimal Control", H. Schulzrinne, S. Casner.

[22] IETF RFC 3388 (December 2002): "Grouping of Media Lines in the Session Description Protocol (SDP)", G. Camarillo, G. Eriksson, J. Holler, H. Schulzrinne.

[23] IETF RFC 2965 (October 2000): "HTTP State Management Mechanism", D. Kristol, L. Montulli.

[24] IETF RFC 3926 (October 2004): "FLUTE - File Delivery over Unidirectional Transport", T. Paila, M. Luby, R. Lehtonen, V. Roca, R. Walsh.

[25] W3C Extensible Markup Language (XML) 1.0, Fourth Edition [Recommendation].

NOTE: Available at http://www.w3.org/TR/2006/REC-xml-20060816/.

[26] Open Mobile Alliance "Rich Media Environment (RME)".

[27] ISO/IEC 14496-4:2004/AMD25 "Information technology - Coding of audio-visual objects - Part 4: Conformance Testing: Amendment 25: LASeR and SAF Conformance".

[28] ISO/IEC 14496-4:2004/AMD27 "Information technology - Coding of audio-visual objects - Part 4: Conformance Testing: Amendment 27: LASeR and SAF Extensions Conformance".

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

DIMS Scene: an SVG scene, which may include extensions, and may be updated over time

DIMS Scene Commands: a set of one or more commands to modify the state of a DIMS Scene

DIMS Unit: the basic unit of transport, processing, and compression, of DIMS content

New Scene: a complete scene (containing an "svg" element), suitable for starting a session or completely replacing the current scene in a session
(Functions very similarly to an I-frame in video)

Normal DIMS Unit: DIMS Units processed when processing a stream (cf. Redundant DIMS Unit)

Primary Stream: a stream which defines the complete scene tree, i.e. in which all random access points are, or build, a complete DIMS Scene

Redundant DIMS Unit: DIMS Units which supply a redundant 'summary' of the stream, and which can be used for random access, tune-in, or error recovery (cf. Normal DIMS Unit)

Scene Update: a set of differences that make changes to the scene in the current session
(Similar to a P-frame in video)

Secondary Stream: a stream which manages only a portion of the scene tree

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

API	Application Program Interface
AVP	Audio/Video Profile
CTR	CounTeR
DIMS	Dynamic and Interactive Multimedia Scenes
DOM	Document Object Model
FLUTE	File deLivery over Unidirectional Transport
HTTP	Hyper Text Transfer Protocol
IANA	Internet Assigned Numbers Authority
ID	IDentifier
LASeR	Lightweight Application Scene Representation
MIME	Multipurpose Internet Mail Extensions
MMS	Multimedia Messaging Service
MTU	Maximum Transmission Unit
PSS	Packet switched Streaming Service
RAP	Random Access Point
RTP	Real-Time transport Protocol
RTSP	Real Time Streaming Protocol
SDP	Session Description Protocol
SMIL	Synchronized Multimedia Integration Language
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
uDOM	microDOM
UDP	User Datagram Protocol
UE	User Equipment
URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	World Wide Web Consortium
XHTML	eXtensible HyperText Markup Language
XML	eXtensible Markup Language

4 Overview and architecture

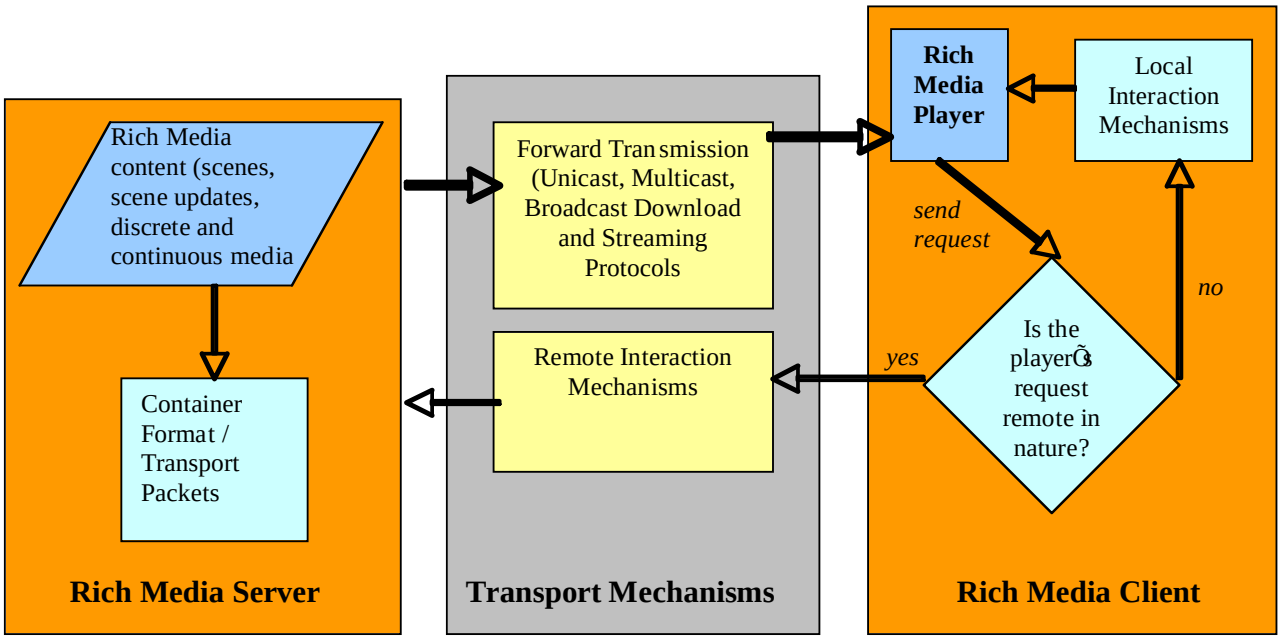


Figure 4-1: General architecture of the rich media system

The rich media system can be perceived as client-server architecture, comprising 3 main components: The rich media server, transport mechanisms and the rich media client. Figure 4-1illustrates the general architecture. The server takes as input, rich media content comprised of scene description, discrete (e.g. images) and continuous (e.g. audio, video) media. Scene description can be dynamically updated through scene updates. The rich media content can be encapsulated into a container format, containing additional information such as media synchronization, metadata, and hint tracks for packetization. The system then utilizes various transport mechanisms for 1-to-1 and 1-to-many protocols for download, progressive download and streaming scenarios. The content is played on the client, allowing for local and remote interactivity of feedback and data requests.

5 Media-type definition

5.1 Introduction

The DIMS media type allows spatial and temporal layout of the multimedia scene. This scene can consist of any combination of still pictures, video, audio, and animated graphics. It includes an update mechanism that allows for partial updates of the existing scene, as well as updating the presentation with a completely new scene and streaming tune-in functionality.

5.2 Media type components

The DIMS media type consists of:

- Base scene description, which is SVG Tiny 1.2 [1].
- Scene description extensions.
- Scene commands.
- Event generation and processing.

5.3 Namespace

The namespace called DIMS here is associated with the URN "http://www.3gpp.org/richmedia/".

5.4 Scene description

5.4.1 Base Scene Description

SVG Tiny 1.2 provides the basic DIMS Scene functionality; layout, inclusion and referencing of objects, synchronization of object timelines and a rendering model.

The full syntax and semantics of SVG Tiny 1.2 shall be supported for DIMS Scene functionality. The version and baseProfile attributes of the SVG element document the version and profile of SVG on which this scene is based.

5.4.2 Scene Description Extensions

5.4.2.1 Introduction

Extensions defined here are designed so that:

a) when the same functionality is present in profiles of SVG other than SVG Tiny 1.2, then the extension is compatible with that or a restricted version of that.

b) A terminal implementing both the present document and SVG (any version) can use a common implementation of the DOM tree, scene graph, rendering model etc. without having variant handling that depends on whether the scene was built using DIMS or SVG.

c) No extensions are required to be present in all documents; content authored to the SVG Tiny 1.2 specification may be used as the initial scene of a stream designed to the present document.

The following extensions are defined here.

5.4.2.2 Rectangular clipping of a graphical object

The lsr:rectClip mechanism provides pixel aligned clipping defined as a transformable rectangle.

The lsr:rectClip element shall be supported. The definition of lsr:rectClip is defined in subclause 6.8.28 of [3].

5.4.2.3 Full-screen video

The full-screen video feature consists of the attribute lsr:fullscreen on the SVG video element.

The lsr:fullscreen attribute shall be supported. The lsr:fullscreen attribute is defined in subclause 6.8.40.2 of [3].

See clause 10 for security considerations of fullscreen.

5.4.2.4 Full-screen SVG

The fullscreen SVG feature in the DIMS namespace consists of an attribute 'fullscreen' on the <svg> element to hint that the scene should be rendered on the full screen. The possible values are "true" and "false" (default). With the attribute set to true the DIMS UE should negotiate the rendering area with its parent UE and get as large part of the screen as possible for the DIMS canvas.

The dims:fullscreen attribute shall be supported on the svg element.

See clause 10 for security considerations of fullscreen.

5.4.2.5 Attributes clipBegin and clipEnd

Attributes clipBegin and clipEnd defined in subclause 7.6.1 of [5] shall be supported on the following elements: video, audio, animation, and the "updates" element as described in subclause 5.4.2.6.

5.4.2.6 Update Streams

The present document defines a new element 'updates' in the DIMS namespace to link secondary streams of updates to a scene. This element has an implicit "simple duration" of 'indefinite'. The synchronization attributes defined in [1] subclause 12.6 can be used with this element.

The dims:updates element shall be supported.

NOTE: lsr:updates defined in subclause 6.8.53 of [3] is a superset of this element.

Attribute definitions:

All timing attributes defined in [1] subclause 16.2.7 are defined for this element, except the "fill" attribute.

The attributes clipBegin and clipEnd defined in subclause5.4.2.5, and syncReference defined in subclause 5.4.2.7, are defined for this element.

xlink:href = "<iri>"

An IRI reference to a source of updates, such as a DIMS stream/file. This attribute specifies the location of the stream of updates. In the absence of this attribute, this element does not have any effect. This attribute is not animatable and not inheritable. In DIMS, support is required only for DIMS streams or files.

5.4.2.7 Synchronization of Media Streams

lsr:syncReference = "<iri>"

The attribute lsr:syncReference from subclause 6.8.8.2 of [3] shall be supported on the elements video, audio, and animation from SVG, and 'updates' from the present document, with the associated synchronization behaviour. This attribute holds a reference to the stream or media element whose clock acts as a clock reference for the stream referred to by this element. This attribute is not animatable and not inheritable.

5.4.2.8 Screen orientation

Two events and two extension strings are defined that make it possible for scenes to adapt to the screen layout. The events are defined in subclause 6.1.3.

Whenever the terminal detects a change of orientation, angle, or screen size, one of these two events is dispatched. A portrait event is dispatched if the screen is taller than it is wide, and a landscape event is dispatched if the screen is wider than it is tall. It is the responsibility of the system below the scene to orient the screen buffer to user; the DIMS Scene author does not do this.

In addition, the orientation is reported in degrees in screenAngle, to the best of the terminal's capability. This is measured as the angle between the positive X-axis of an un-rotated frame of reference, and the orientation of the longer of the positive X or Y axis of the screen, as rotated, as shown in Figure 5-1. Note that the SVG Y axis is downward.

Specifically, for a screen that is normally portrait and in its normal position, the screenAngle is 90 degrees, since the longest axis is vertical, and the Y-axis is downward in SVG. Similarly, for a terminal that is normally landscape and in its normal position, the screenAngle is 0 degrees, since the longest axis is horizontal, and is the X-axis. This angle therefore would normally be close to 90 or 270 in portrait events, and close to 0 or 180 in landscape events, and 0 or 90 in terminals that are in their normal orientation, and 180 or 270 in terminals that are inverted.

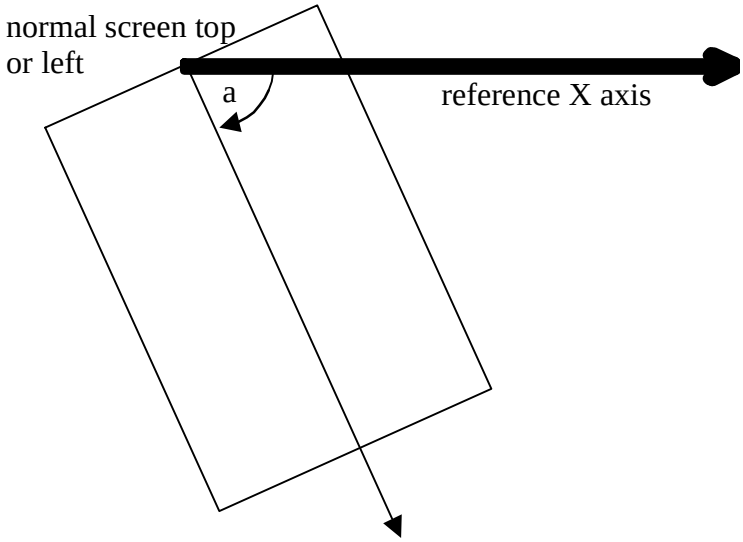


Figure 5-1: Screen Orientation

These events shall map to the ScreenOrientationEvent interface as defined in subclause 6.1.3.

The screen orientation events shall be supported in DIMS. If the terminal has an orientation sensor, or other physical adaptation that causes the available screen drawing area to change (e.g. a partial cover), events shall be generated whenever the terminal detects a change in any of the parameters to these events.

The following extension strings shall also be supported, in order to allow the use of the switch element:

- orientLandscape for typical 'landscape' orientation;
- orientPortrait for typical 'portrait' orientation.

The namespace of these feature strings is DIMS.

If the most recent event generated was a portrait event, then `requiredExtensions="orientPortrait"` tests as true; if the most recent event was a landscape event, `requiredExtensions="orientLandscape"` tests as true. At any time, exactly one of these requiredExtensions expressions shall test as true. If no event has been generated, the appropriate requiredExtensions expression tests as true.

Information on softkey location and key mapping may be found in section 9 of [26].

5.4.2.9 Current-Time Indication

In a Primary Stream, Redundant Random-Access Point there is a need to establish the current SceneTime of the scene, so that terminals tuning-in, performing random-access, or recovering from a lost high-priority DIMS Unit achieve the same SceneTime as terminals which had processed the entire stream from the most recent non-redundant Random Access Point. This media-time (scene time) is indicated by the currentSceneTime attribute on the SVG element, and takes a valid clock value in the document timeline, from the SVG specification [1].

The scene state is set exactly as if the SVG document had been loaded and displayed at the non-zero time T in the current-time indicator.

EXAMPLE: This is the same as if this SVG scene had been used in a SMIL document as the target of an "animation" element with clipBegin of T, or if conceptually all absolute times S in the document were replaced with S-T and the document instantiated at time 0.

Attribute definition:

currentSceneTime = "<clock-value>"

Specifies the current scene time (a valid clock value) in the document timeline, at which the scene is displayed. The scene state is set exactly as if the SVG document had been loaded and displayed at the non-zero time T in the current-

time indicator. This attribute is defined in DIMS namespace, and may be present on the root SVG element in redundant random access points. The default value is zero.

5.4.2.10 Active attribute

On all elements, the following attribute is defined in the DIMS namespace:

`active`: this attribute defines whether the element is active. The possible values are "true" (default) and "false".

Setting the value of this attribute to true or false is equivalent to executing the commands activate and deactivate. See subclause 5.5.3 for the behaviour of deactivated elements. This attribute is not animatable and not inheritable.

The dims:active attribute shall be supported.

5.5 Scene Commands

5.5.1 Scene Updates

The scene update mechanism allows reception of updates that change parts of the current scene, without having to replace the entire scene.

To account for the different update scenarios two update mechanisms are defined:

- **Primary-stream updates:** Updates are delivered to the client in the same stream as the original scene.
- **Secondary-stream updates:** Updates are delivered to the client in separate streams from the original scene, e.g. in an interactive scenario or initiated from the scene mark-up.

In a primary-stream case, the updates and/or scene replacements are sent in the same stream as the initial scene. The temporal management of samples in a primary stream is based upon transport level timestamps. A secondary stream is a stream that does not contain the initial scene. A secondary stream is initiated directly from the DIMS mark-up using the 'updates' element.

The following LASeR commands from subclause 6.7 of [3] in LASeR ML format shall be supported.

- The LASeR Insert command from subclause 6.7.5 of [3] shall be supported on elements, attributes and values in list attributes with the following relaxed constraints: values *may* be inserted on attributes x and y of the text element.
- The LASeR Delete command from subclause 6.7.4 of [3] shall be supported on elements, attributes and values in list attributes.
- The LASeR Replace command from subclause 6.7.8 of [3] shall be supported on elements, attributes and values in list attributes with the following relaxed constraints: attributes attributeName, id, type, xml:space, preserveAspectRatio and the x and y attributes of the text element *can* be replaced. There are no restrictions on the value of attributeName. The text regarding executionTime does not apply.
- The LASeR Add command from subclause 6.7.2 of [3] shall be supported.

5.5.2 State management commands

The following state management commands shall be supported:

- The LASeR Save command from subclause 6.7.10 of [3] shall be supported.
- The LASeR Restore command from subclause 6.7.9 of [3] shall be supported.
- The LASeR Clean command from subclause 6.7.3 of [3] shall be supported.

These LASeR commands are defined as an interface to persistent storage. Selected scene information is cached on a best effort basis. The security principles behind this caching are those of the state caching mechanism in HTTP, commonly called cookies [23].

The saved data is defined by a groupID (known as "name" in [23]) and scoped by the "service" defined by a domain-name and path; for a command to operate on the data, all must match.

The LASeR command "save" saves the values of a set of attributes, each identified by element ID and attribute name. Each save operation uses a groupID. Any other saved state with the same domain-name, path, and groupID is replaced.

The LASeR command "restore" restores the attributes (if any) previously saved and scoped by the domain-name and path. The set of data restored is defined below.

The LASeR command "clean" erases the attributes (if any) previously saved and scoped by the domain-name and path. The set of data erased is defined below.

The following two attributes are defined in the stream signalling, and define the security restrictions for the above commands:

- ***useFullRequestHost***: this Boolean attribute indicates whether the full domain-name of the request-host is used (true, 1) or the first component of the domain-name is elided (false, 0). For example, if the source material came from "www.example.org", then this differentiates between associating the "service" with "www.example.org" and ".example.org". (Note the definition of local names in [23], and the possibility to associate the "service" with locally loaded files, and that the domain-name may be either "<hostname>.local" or ".local" in that case.)

- ***pathComponents***: this 4-bit unsigned integer attribute indicates how much of the source path is used. If this takes the value 0, then the "service" is not associated with a path, and if it takes the special value 15 (or any value equal to or greater than the number of components in the path) then the entire path is used up to but excluding the final file-name. For example, if the source was "/user/laser-expert/demo/art.mp4" then a value of 3 or greater selects "/user/laser-expert/demo" as the path, the value 2 selects "/user/laser-expert" and the value zero sets no path.

Data is saved as a set of four values, using the URI, pathComponents and useFullRequestHost from the stream containing the save command:
- the domain-name formed from the URI and useFullRequestHost;
- the path formed from the URI and pathComponents;
- the groupID;
- the set of {element-ID, attribute-name, value} triplets.

When a restore command is executed all saved sets with the same (equal) groupID, and also where the URI of the stream containing the restore command matches the saved domain-name and path, are restored. This matching is defined in section 3.3.4 of [23].

A clean command behaves exactly the same as a save command that saves no state; as is normal for the save command, any other saved state with the same domain-name, path, and groupID is replaced, in this case, with an empty set of saved data. This is functionally equivalent to deleting that saved state, as nothing would be restored.

NOTE: Be aware that though the data saved and restored is scoped by stream, once it is restored into the tree it is globally visible.

5.5.3 Activate and Deactivate

The commands activate and deactivate as defined in subclauses 6.7.12 and 6.7.13 of [3] shall be supported, in a manner that is functionally equivalent to that specification. These commands have one attribute:

- ***ref***: the id of the element which is to be activated or de-activated.

When an element is deactivated, the system then treats the DOM tree as if that element and its descendents were not present in the DOM tree, and invisible to everything except commands and scripts. Commands and scripts can reference it as if it were still in the DOM tree. When activated, the element is then restored to visibility, in the same location in the tree as if it had not been previously deactivated.

The active attribute is not inherited. However, activated and de-activated events are generated if the effective active state of an element changes as a result of a change to the active state of a parent. This means that a deactivated event may occur for an element, as a result of deactivating a parent, when a test of its dims:active attribute value returns true.

5.5.4 Distributed Random Access Points

5.5.4.1 Introduction

A Distributed Random Access Point (DRAP) is a redundant DIMS tune-in point (either primary or secondary) that can, instead of explicitly defining all elements itself, reference elements in coming DIMS units. The commands in these following DIMS units are not executed, elements are simply copied according to references in the DRAP. These references can be used to reduce redundancy (i.e. not defining an element both in a RAP and an update) or to simply spread the size of the RAP over a period of time.

After this copying operation, the pending action(s) in the DRAP are complete, and are then executed, and normal processing resumes.

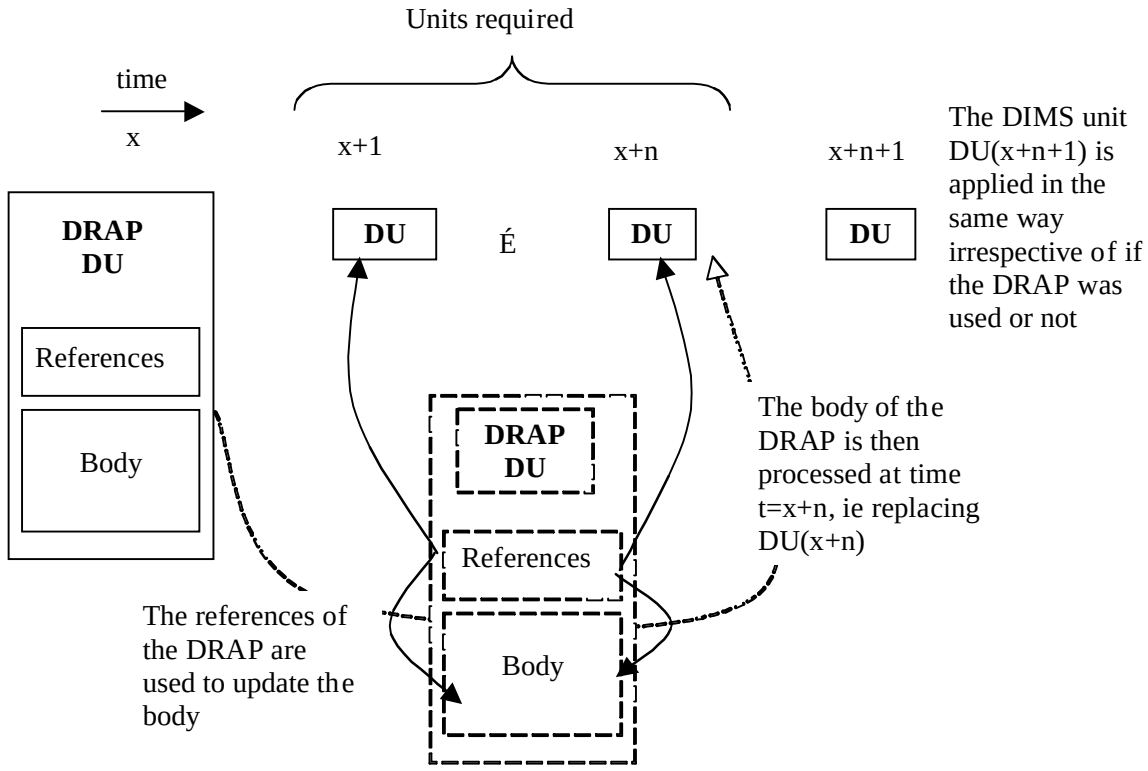


Figure 5-2: Illustration of the DRAP Concept

5.5.4.2 DRAP syntax and semantics

The rootmost element in a DRAP document shall be a <drap> element in the DIMS namespace. A DIMS Unit containing a DRAP shall contain only the DRAP.

Attribute definitions:

unitsrequired="*units-required*"

Indicates the number of coming DIMS units required.

NOTE 1: These DIMS units are *not* executed in the normal way when tuning in using DRAP; instead, they are used as needed as a source of material for the DRAP.

The DRAP element contains one or more getfromupdate elements, which form the processing instructions, and one or more other elements that form the pending action(s). The processing instructions are applied to the pending action. The indicated number of DIMS units are processed for the DRAP, and the pending action(s) are performed at the time of the DIMS unit at the indicated distance, and normal DIMS unit processing resumes.

NOTE 2: All the getfromupdates should have been resolved by the indicated distance.

The getfromupdate element shall reference an element in another DIMS unit and an element in the pending actions. The element referred to in the other DIMS unit shall replace the element in the DRAP pending actions.

Attribute definitions:

source="*elementid*"

Specifies an xml id appearing in an upcoming DIMS unit. If the same xml id appears in different DIMS units, it shall not make a difference which one the client chooses.

target="*elementid*"

Specifies an xml id appearing in the DRAP pending action(s).

5.5.5 Immediate Script Execution

The doScript command in the DIMS namespace shall be supported. This command supplies a script for immediate execution, including the ability to update the DOM. It has a single attribute, the type of the script. The script is in the body of the element. Processing this command involves executing the script in the context of the DIMS stream in which it occurs.

Attributes:

`type` - is a string that identifies the scripting language used. It takes a suitable MIME type [18] from the IANA registry, such as "application/ecmascript" (see [13]).

An example is:

```
<doScript type="application/ecmascript">
    var root = document.documentElement;
    var myGroup = document.createElementNS(
        "http://www.w3.org/2000/svg", "g");
    myGroup.id = "myGroup";
    root.appendChild(myGroup);
    var myRect = document.createElementNS(
        "http://www.w3.org/2000/svg", "rect");
    myRect.id = "myRect";
    var color = root.createRGBColor( 255, 0, 0);
    myRect.setRGBColorTrait("fill", color);
    myRect.setFloatTrait("x", 10);
    myGroup.appendChild(myRect);
</doScript>
```

5.5.6 Seeking in the DIMS Stream

<seek seekOffset="*seekOffset*" />

Attributes:

`seekOffset`: A clock value from subclause 16.2.7 of [1].

The command seek in the DIMS namespace results in a seek, by the amount seekOffset, in the DIMS stream timeline. The target stream time is obtained by adding seekOffset to the current stream time. As a DIMS stream may contain multiple scenes, this seeking can result in a change of scene. A seek to the local time corresponding to the seekOffset shall be applied to the (possibly new) scene. The seek command shall be supported.

NOTE: Seeking can be conceptually seen as a function where the global timeline and document timelines are moving forward in a synchronized manner, just as in normal playback, but more quickly and without rendering. A seek backwards in time (negative seekOffset) could be done in a similar way, but by starting again from zero and moving forward.

5.6 DIMS Unit Definition

5.6.1 Definition

A DIMS Unit is built from a header and a body. The DIMS Unit Body is either:

a) a complete SVG document as specified in subclause 5.4, possibly using extensions; or

b) a textually concatenated sequence of scene commands as specified in subclause 5.5;

A DIMS Unit Body may be compressed.

DIMS Units are framed by the transport layer. Each DIMS Unit has certain characteristics, signalled by the DIMS Unit Header.

There are DIMS Units used in redundant processing, and DIMS Units used in normal processing. Redundant DIMS Units, and DIMS Units marked as random-access points, are used in random access, tune-in, and error recovery; for a full description of their processing model, see subclause 5.8.

5.6.2 DIMS Unit Header

DIMS Unit Header is 1 byte long. The length of a DIMS Unit is the length of the DIMS Unit Body plus the length of the DIMS Unit Header. DIMS Unit lengths are carried by the transport layer.

The DIMS Unit Header has the following layout.

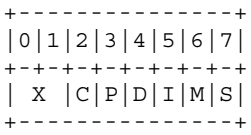


Figure 5-3: DIMS Unit header

These fields have the following definitions:

S: is-Scene: when 1, indicates that the DIMS Unit contains a Scene Description as documented in subclause 5.4; when 0, indicates that the DIMS Unit contains one or more Scene Commands as documented in subclause 5.5.

M: is-RAP: when 1, indicates a Random Access Point; when 0, indicates a non-Random-access point.

I: is-redundant: when 0, indicates a main (normal processing) DIMS Unit; when 1, signals a redundant DIMS Unit.

D: redundant-exit: shall be 0 on DIMS Units with is-redundant==0; on DIMS Units with is-redundant==1, when 1, indicates that redundant processing is completed by this DIMS Unit, and normal processing should begin, and when 0, indicates that redundant processing should continue.

P: priority set to 1 indicates a high-priority unit; when set to 0 indicates a low-priority unit. A unit marked as low-priority indicates that if the unit, or any sequence of such low-priority units, is not processed by the terminal:

1. all succeeding DIMS Units can be decoded and operated on without error (e.g. their DOM updates do not depend on the possibly lost command(s).)
2. the visual and semantic nature of the scene is satisfactory to the content author.
3. The loss of those units does not require the terminal to enter into a tune-in or repair state.

Units fulfilling the above criteria should be marked as low-priority, and shall be marked as high-priority otherwise.

DIMS Units with is-redundant set to 1 should normally be marked as low-priority, to avoid their loss causing an un-needed entry into tune-in state when redundant and normal data are carried in the same transport.

C: compression: indicates the compression applied;
0 indicates no compression (textual format);
1 indicates that the content is compressed using the encoding signalled in stream setup.

X: reserved: shall be set to 0 and shall be ignored by the receiver

NOTE: The setting of the priority field is, due to point 2 above, partly at the discretion of the content creator. An example of a simple method of evaluating point 2 is to see if, when the next packet is received, the terminal state is identical to what it would have been if the DIMS Unit(s) had not been lost in the first place.

5.7 Timing model

DIMS inherits the timing model from [1] in its entirety. This section defines the timing of DIMS units.

DIMS units are associated with a media timestamp which marks

a) the time when the SVG scene is instantiated and the scene time resets to 0, for New Scenes, or

b) the time when DIMS units are applied and affect the scene, for other DIMS units.

The processing order for scene updates is the same as for script and event processing. The present document does not mandate any relative timing or processing order for DIMS units, scripts or events that shall be processed at the same scene time. DIMS Units from the same stream shall be processed in decoding order, i.e. sequence number order in RTP or order inside a sample in the 3GP file format.

In the main stream the SVG time has periodic resets to 0 on each New Scene (what SVG calls "document time" and is called here "SVG Scene Time"). The SVG scene time, and the media timestamps of the DIMS presentation, are related by a piecewise linear relationship, with discontinuities only at the New Scenes. Secondary streams are synchronized to the main stream using the normal mechanisms for media streams.

DIMS Units containing commands, with the same timestamp in the same stream, are applied "instantaneously" in scene time, that is, the scene time does not change while they are being applied.

DIMS units containing commands are applied atomically and have exclusive access to the scene tree during processing; only events and scripts directly resulting from the processing of the DIMS unit, are run.

5.8 Processing Model

A Scene Description is processed as a complete replacement for the current scene tree. That is, the entire DOM is discarded and replaced with the result of parsing the SVG element. All other DIMS Units retain (and possibly modify) the current scene tree.

All high-priority data units not marked as redundant shall be processed during normal decoding. All low-priority data units not marked as redundant should be processed during normal decoding. Data units marked as redundant should be ignored during normal processing. All data units marked as RAP are suitable tune-in points. When tuning-in to a stream, decoding shall begin by, at the latest, the first unit marked as RAP irrespective of the value of its is-redundant flag.

If a normal (non-redundant) random access point is identified during redundant processing or DRAP processing, the normal random access point should take precedence.

Commands that cannot be executed (e.g. they refer to a DOM node which does not exist) shall be ignored when in tune-in or redundant-processing. This condition should not arise in normal processing, and their handling in this state is not defined by the present document.

The following state diagram, and processing pseudo-code for each state, illustrate the states and the use of the various flags in the DIMS unit header, and comply with the mandatory and recommended processing requirements. The state diagram and pseudo-code, or better, should be implemented in DIMS clients.

In the state diagram and pseudo-code, the terminal may be processing a stream under one of three conditions:

a) normal processing, 'normal';

b) after tuning in, performing random access, or when loss is detected, 'tune-in';

c) while processing redundant DIMS units, 'process-redundant'.

Tune-in state is entered under any of the following circumstances:

a) after opening a stream;

b) after performing random access;

c) after loss of a high-priority DIMS Unit in normal processing;

d) after loss of any DIMS Unit in redundant processing.

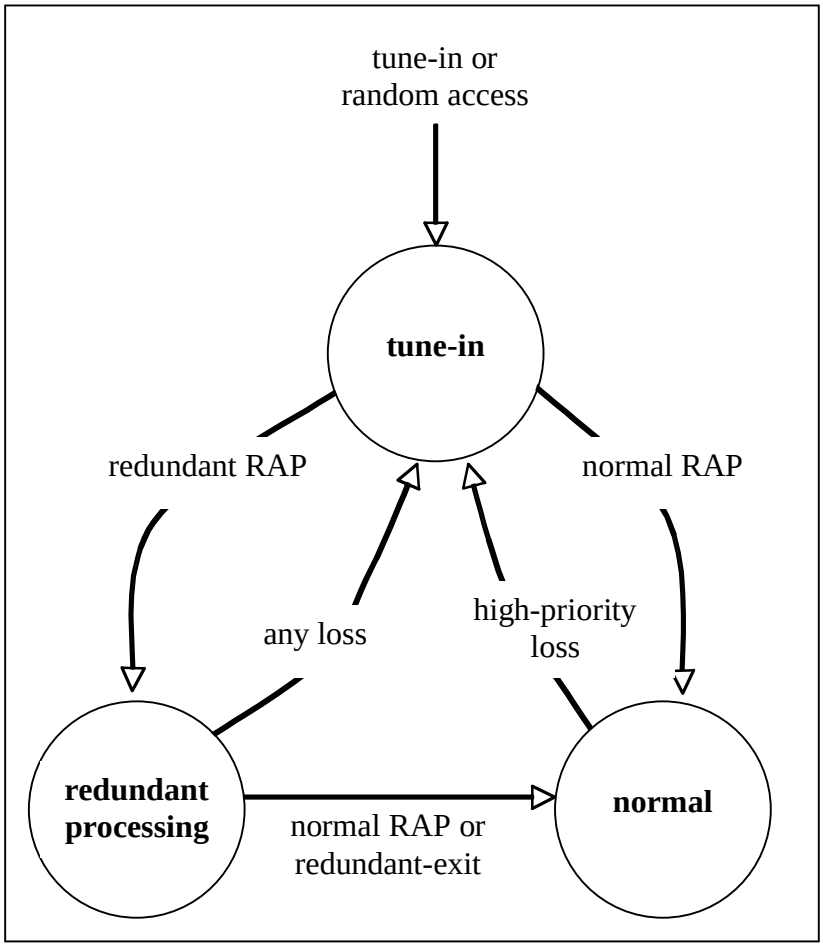


Figure 5-4: DIMS Client State Diagram

The following behaviour is performed for each DIMS Unit in each state, and then a state transition is performed as indicated by the state diagram.

```
Normal state:
    if DIMS-Unit.is-redundant
        then Discard(DIMS-Unit)
        else Process(DIMS-Unit);

Tune-in state:
    if DIMS-Unit.is-RAP
        then Process(DIMS-Unit)
        else Discard(DIMS-Unit);

Redundant-processing:
    if (     DIMS-unit.is-redundant) ||
```

```
((not DIMS-Unit.is-redundant) && DIMS-unit.is-RAP)
 then Process(DIMS-unit)
 else Discard(DIMS-Unit);
```

Where this pseudo-code indicates that a DIMS Unit is processed, then if a Distributed Random Access Point (DRAP) is in process, elements required by the DRAP are extracted from this unit. If a DRAP is not in process, the DIMS Unit is processed as normal. If one of the DIMS units identified by units-required is a normal (non-redundant) random access point, DRAP processing should be abandoned, and that normal RAP processed in the usual way.

5.9 Random Access, Tune-in and Error Recovery

5.9.1 Introduction

Random access points in streams are either normal random access points or redundant random access points. Normal random access points are processed by terminals in all states. Redundant random access points should only be processed only by terminals needing to perform random access, tune-in, or error recovery.

Random access points are indicated in the DIMS Unit header using the is-RAP flag. Redundant random access points have the is-Redundant flag set to 1; normal Random Access points have this flag set to 0.

5.9.2 Random Access Points in Primary Streams

A Random Access Point (normal or redundant RAP) in a primary stream shall either contain an entire scene (i.e. be a Scene Description) or the mechanism to build an entire scene (such as DRAP). When used, this scene becomes the current scene and replaces all previous data. There may be further DIMS Units with the same timestamp that modify the scene tree.

A redundant Random Access Point in a primary stream shall have the currentSceneTime attribute on the SVG element. Any following commands in subsequent DIMS Units with the same timestamp are processed at this time.

5.9.3 Random Access Points in Secondary Streams

A Random Access Point (normal or redundant RAP) in a secondary stream shall either contain an entire update (i.e. a series of commands) or the mechanism to build an entire update (such as DRAP). The command(s) provided set the scene (specifically, the portion of the scene managed by the secondary stream) into an appropriate state, whether the random access point is used for initial tune-in, or for error recovery.

NOTE: The secondary stream needs to be encoded in such a way that it does not matter which packets were lost or this is an initial tune-in or random access - the appropriate state is set. This would include removing any elements or attributes which should have been removed, etc. A simple way of encoding such a stream would be to only let updates in a secondary stream make modifications to a few nodes. Then this operation could be as simple as removing these few nodes and reinserting them, removing all potential errors.

5.9.4 Error Recovery

There are several error resilience mechanisms available in DIMS. Among these are:

- Priority: By separating essential and non-essential units one can determine if a loss need repair or not. This is described in subclause 7.3.1.
- Periodic Random Access Points (RAPs): Random Access Points can be placed periodically in a stream. In the case of error one can tune-in to the channel again.
- Separation of static and dynamic data. This can even increase the efficiency of Distributed Random Access Points.

A combination of these methods can be used.

6 Interaction and Scripting

6.1 Local interaction

6.1.1 DOM Level 3 events

The supported local events and their management in DIMS are built upon the events model described in [1].

They include DOM Events (focus, activate, etc.), SVG Events (connection, load, etc.) and general XML events [19] (user events, timing, key, and pointer events).

6.1.2 Media Access Events

The media access events defined in [20] shall be supported.

6.1.3 Screen Orientation Events

The following events shall be supported.

Event Type	Namespace	Description	Interface	Bubble	Canc
"screenOrientationPortrait"	DIMS	The screen orientation has changed to typical 'landscape' orientation	ScreenOrientationEvent	No	No
"screenOrientationLandscape"	DIMS	The screen orientation has changed to typical 'portrait' orientation	ScreenOrientationEvent	No	No

```
interface ScreenOrientationEvent : Event
{
    readonly attribute unsigned long screenWidth;
    readonly attribute unsigned long screenHeight;
    readonly attribute unsigned long screenAngle;
}
```

screenWidth - contains the new screen display or viewport width.

screenHeight -contains the new screen display or viewport height.

screenAngle - documents the angle between the primary axis of the screen, and an unrotated horizontal axis (see 5.4.2.8) with a value between 0 and 359 inclusive.

NOTE: A superset of the ScreenOrientationEvent interface is specified in [26].

6.1.4 Other Events

The events pausedevent and resumedevent from subclause 6.5.2 of [3] shall be supported.

The following events shall be supported.

Event Type	Namespace	Description	Interface	Bubble	Canc
"activatedEvent"	DIMS	Occurs when an element changes state from deactivated to activated	Event (Annex A of [1])	No	No
"deactivatedEvent"	DIMS	Occurs when an element changes state from activated to deactivated	Event (Annex A of [1])	No	No

6.2 Remote interaction

Client-server communication is possible in the DIMS system using three different mechanisms:

- The client can open a suitable URL. The set of valid URL forms is not specified in DIMS, and may include, for example, protocols such as HTTP [12], RTSP [16] or MailTo.
- By establishing a socket connection between the client and the server using the Connection API in the uDOM [17].
- By using the HTTP specific SVG uDOM methods getURL or postURL [17].

6.3 Scripting

SVG Tiny 1.2 contains a uDOM interface that provides linkage to a script engine and adds the possibility to modify the DOM representation of the scene from scripts.

ECMAScript mobile profile (MP) [2] can be used in conjunction with the script and handler elements and SVG µDOM API (Appendix A of [1]) in order to provide more powerful DOM manipulation, and interaction.

UEs supporting the DIMS media type shall support ECMAScript mobile profile (MP) [2] with the following extensions to uDOM API.

Table 1 adds to the table in subclause A.8.12 of [1]. It contains trait access rules for DIMS extensions.

Table 1: Trait access rules for DIMS extensions

Attribute	Trait Getter	Trait Setter	Default Values	Description
lsr:fullscreen	getTraitNS[true | false]	setTraitNS[true | false]	false	Available on <video> element
dims:fullscreen	getTraitNS[true | false]	setTraitNS[true | false]	false	Available on the <svg> element
lsr:x	getFloatTraitNS	setFloatTraitNS	0.0f	Origin x of the <rectClip>
lsr:y	getFloatTraitNS	setFloatTraitNS	0.0f	Origin y of the <rectClip>
lsr:width	getFloatTraitNS	setFloatTraitNS	0.0f	Width of the clipping region defined by <rectClip>
lsr:height	getFloatTraitNS	setFloatTraitNS	0.0f	Height of the clipping region defined by <rectClip>
smil:clipBegin	getTraitNS	setTraitNS	""	Available on <video>, <audio>, <animation>, and <updates> elements
smil:clipEnd	getTraitNS	setTraitNS	""	Available on <video>, <audio>, <animation>, and <updates> elements
dims:active	getTraitNS[true | false]	setTraitNS[true | false]	true	Available on all SVG elements

Description of getFloatTraitNS and setFloatTraitNS methods available on the TraitAccess interface:

float getFloatTraitNS(in DOMString namespaceURI, in DOMString name) raises(DOMException);

- Same as getFloatTrait, but for namespaced traits. Parameter name shall be a non-qualified trait name, i.e. without prefix.

Parameters:

`namespaceURI` - the namespaceURI of the trait to retrieve.
`name` - the name of the trait to retrieve.

Return Value:

the trait value as float.

Exceptions:

DOMException - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.

DOMException - with error code TYPE_MISMATCH_ERR if requested trait's computed value cannot be converted to a float.

void setFloatTraitNS(in DOMString namespaceURI, in DOMString name, in float value) raises(DOMException);

Same as setFloatTrait, but for namespaced traits. Parameter name shall be a non-qualified trait name, i.e. without prefix.

Parameters:

namespaceURI - the namespaceURI of the trait to be set.
name - the name of the trait to be set.
value - the value of the trait to be set as float.

Exceptions:

DOMException - with error code NOT_SUPPORTED_ERR if the requested trait is not supported on this element or null.

DOMException - with error code TYPE_MISMATCH_ERR if the requested trait's value cannot be specified as a float (for e.g. NaN)

DOMException - with error code INVALID_ACCESS_ERR if the input value is an invalid value for the given trait or null.

ECMAScript bindings for the extensions defined in this specification can be found in Appendix D of [26].

7 Transport

7.1 Overview

The transport mechanisms support rich media delivery in the following modes: Unicast download (HTTP/TCP [12] or MMS [6] protocol), broadcast/multicast download (FLUTE/UDP [24]), unicast streaming and broadcast/multicast streaming (RTP/UDP [15]). For download mode, reliability is guaranteed by existing mechanisms in the transport and network layers, and no error resilience tools need to be designed at the application layer for rich media delivery. However, rich media transport in streaming mode is more challenging, with UDP being unreliable. Therefore, the RTP design provides some error resilience tools to help the media decoder cope with unreliable transport.

Rich media is a combination of continuous media and discrete media and relevant transport mechanisms for these two media types should be used. Rich media streaming is thus naturally realized by:

a) streaming continuous media such as scene streams, video and audio; and

b) downloading the discrete media, such as images.

DIMS Units can be classified as either used in normal processing, or used only for 'redundant' processing. For a given DIMS data-stream, these two kinds of DIMS Units can be managed either:

a) in a single transport; or

b) in two separate transports.

7.2 Storage in ISO Base Media File Format Files

7.2.1 Introduction

DIMS streams, both primary streams (those containing SVG scenes) and secondary streams (which normally carry only updates) are carried in files of the ISO Base Media File Format [10] (including 3GP files [8]) according to this subclause.

Either one or two tracks are used in the file for the normal and redundant DIMS Units.

7.2.2 Stream Type

Scenes are carried in scene tracks in ISO family files. They therefore use:

a) a video media handler 'vmhd';

b) a media handler type of 'sdsm' (scene description media handler);

c) a derivative of the base SampleEntry in the sample description box.

The timescale for the stream should be suitably chosen to achieve the desired accuracy of timing.

7.2.3 Track and Media Header fields

The width and height in the track header shall be set in the desired ratio, and indicate the suggested minimum display size. A player on a system with an indefinitely large display, in the absence of a fullscreen request, could use this size as a suggested initial display size.

If the presentation has an expected, reasonable duration, then it is encoded as the track duration. Otherwise the ISO file format recommendation of maxint for the duration, when it is indeterminate, should be used.

The language code of the track should be set appropriately if the presentation is language-specific, or else the value 'und' (undetermined) or 'mul' (multiple) should be used.

7.2.4 Sample Dependency Table

The sample dependency table may be used. The 'unknown' field values may be needed under some circumstances. The fields have the following semantics for DIMS streams:

`sample_depends_on` should be set according to whether the sample contains a normal DIMS Unit (not is-redundant) with is-RAP set to 1:

0: unknown;

1: this sample does not contain a normal RAP;

2: this sample does contain a normal RAP;

3: reserved.

`sample_is_depended_on` should be set according to the value of the P-bit in the DIMS Unit headers:

0: unknown;

1: one or more DIMS Units have the P-bit set to 1;

2: no DIMS Unit has the P-bit set to 1 (low-priority sample);

3: reserved.

`sample_has_redundancy` should be set to indicate whether the sample contains redundant DIMS Units:

0: unknown;

1: one or more DIMS Units have the is-redundant bit set to 1;

2: no DIMS Unit has the is-redundant set to 1;

3: reserved.

7.2.5 Sample Entry Name and Format

The sample entry four-character code for scenes is 'dims'. The configuration box shall be present in the sample entry.

```
class SceneConfiguration extends FullBox ('dimC', version = 0, 0){
    unsigned int(8) profile;
    unsigned int(8) level;
    unsigned int(4) pathComponents;
    unsigned int(1) useFullRequestHost;
    unsigned int(1) stream_type;
    unsigned int(2) contains_redundant;
    string      text_encoding;
    string      content_coding;
}
class MPEG4BitRateBox extends Box('btrt'){
    unsigned int(32) bufferSizeDB;
    unsigned int(32) maxBitrate;
    unsigned int(32) avgBitrate;
}

class DIMSScriptTypes extends Box('diST')
{
    string      content_script_types;
}

class DIMSSampleEntry() extends SampleEntry ('dims'){
    SceneConfiguration      config;         // mandatory
    DIMSScriptTypes         scripts;        // optional
    MPEG4BitRateBox         bitrateinfo;    // optional
}
```

The fields have the following semantics:

- `profile` – Specifies the profile of DIMS used, for example the valued indicating Mobile Profile as defined in section 8.1
- `level` – Specifies the minimum DIMS level needed to be able to display the scene as defined in section 8.2.
- `stream_type` - takes the value 1 for primary streams, and the value 0 for secondary streams. Files containing secondary streams are not normally playable by themselves, outside the context of the scene(s) they are designed to update.
- `contains_redundant` - takes the value 1 ("main") if the stream contains only DIMS Units with is-redundant set to 0, the value 2 if the stream contains only DIMS Units with is-redundant set to 1 ("redundant"), and takes the value 3 ("main+redundant") if both occur. The value 0 is reserved. Note that streams containing only redundant units must be linked to the main stream for which they are redundant (see subclause 7.2.9).
- `text_encoding` - is a null terminated string with possible values taken the XML specification for character encoding in entities (e.g. subclause 4.3.3 of XML 1.0 Fourth edition [25]). It describes the text encoding after the content has been de-compressed (e.g. after deflating). This field is only applicable if the content is transmitted as (possibly encoded) text. An empty string shall be used otherwise.
- `content_coding` - this field provides the identification of the compression scheme. It is a null terminated string specifying the encoding (compression) format of the content. It is defined in the same way as the content-coding header in HTTP (subclause 3.5 of [12]). An empty string indicates that no compression is used.
- `content_script_types` - is a null terminated string that identifies the scripting languages used. It is a comma-separated list of MIME types [18] from the IANA registry, such as "application/ecmascript" (see [13]). It shall provide a complete listing of the script types that the terminal must support in order to process the stream. If the box is not present, the set of required script types is unknown. If the box contains the empty string, then the stream does not require any script processing.

- `bufferSizeDB` gives the size of the decoding buffer for the elementary stream in bytes. This is the size of the largest buffer needed to hold a sample in textual format, in bytes (i.e. after any de-compression).
- `maxBitrate` gives the maximum rate in bits/second over any window of one second.
- `avgBitrate` gives the average rate in bits/second over the entire presentation.
- `useFullRequestHost` and `pathComponents` are defined in subclause 5.5.2.

The text_encoding is required to be consistent over all the DIMS units described by this sample entry. This simplifies processing. It is an error to have a mismatch between this value and those present in the XML of the DIMS units themselves.

7.2.6 Sample Format

A sample is a concatenated sequence of one or more DIMS Units associated with the same media time, with a two-byte length field in network (big-endian) format preceding each DIMS Unit. The length is the length of the DIMS Unit not including the length field itself (that is, the combined length of the DIMS Unit Body and DIMS Unit Header).

A sample may contain one or more Normal DIMS Units or Redundant DIMS Units, or both, associated with the same media time.

7.2.7 Other Resources

Other resources may be carried in the meta-data directories of ISO files, in the track containing the scene, the movie containing that track, or the file containing that movie. If there is no actual meta-data (the meta-data block is there merely to carry resources), the meta-data handler type 'null' may be used.

URL forms to address these resources are defined in the ISO specification, and are relative to the file containing the resource.

The meta data box may also be used for multi-scene presentations where the meta box includes the initial SVG scene, and one of the tracks provides the updates.

7.2.8 Sync Samples

The sync sample table marks samples in which any of the DIMS Units have the is-RAP bit set to 1.

NOTE: The use of the shadow sync box is deprecated.

7.2.9 Separate Redundant Track

Redundant DIMS Units may be stored in the file format using a separate track. The redundant track shall be linked to the matching main track by a track reference of type 'swto' in the redundant track.

Redundant tracks are identified by this track reference, and shall also have contains_redundant set to "redundant data only" in their sample entry. The track they link to shall have contains_redundant set to " main data only".

If a stream is converted from a single-track to two-tracks, some small adjustment may be needed. Specifically, any 'normal' DIMS Units following the redundant-exit indication in the same sample will need to be copied into the 'redundant' track, marked as 'redundant' DIMS Units, and the redundant-exit indication moved to the last such DIMS Unit in the sample.

A terminal may perform tune-in etc. using the 'redundant' track by:

a) finding the random access point in the redundant track, closely preceding the desired play point, by using the sync sample table;

b) processing DIMS Units from the redundant track until the redundant-exit indication;

c) following the 'swto' track reference and commencing processing at the temporally next sample in the linked (main) track.

7.3 RTP Payload format for DIMS Streams

7.3.1 Priority

The counter (CTR) field is used to detect the loss of high priority DIMS units. Encoders and decoders keep a running value of the counter; the encoder places in each packet the current value of the counter; after being placed in the packet, the running counter is incremented by one if that packet contains one or more DIMS Units with high priority. The decoder compares the CTR field of each incoming packet with its running counter, and thereby checks for high-priority loss. After the check, the decoder's running counter is incremented by one if the received packet contains one or more DIMS Units with high priority.

NOTE: A discontinuity in the sequence number indicates a lost packet. A discontinuity in the CTR field indicates the number of prioritized packets which have been lost.

An example of the use of the CTR and priority (P) bits is shown below:

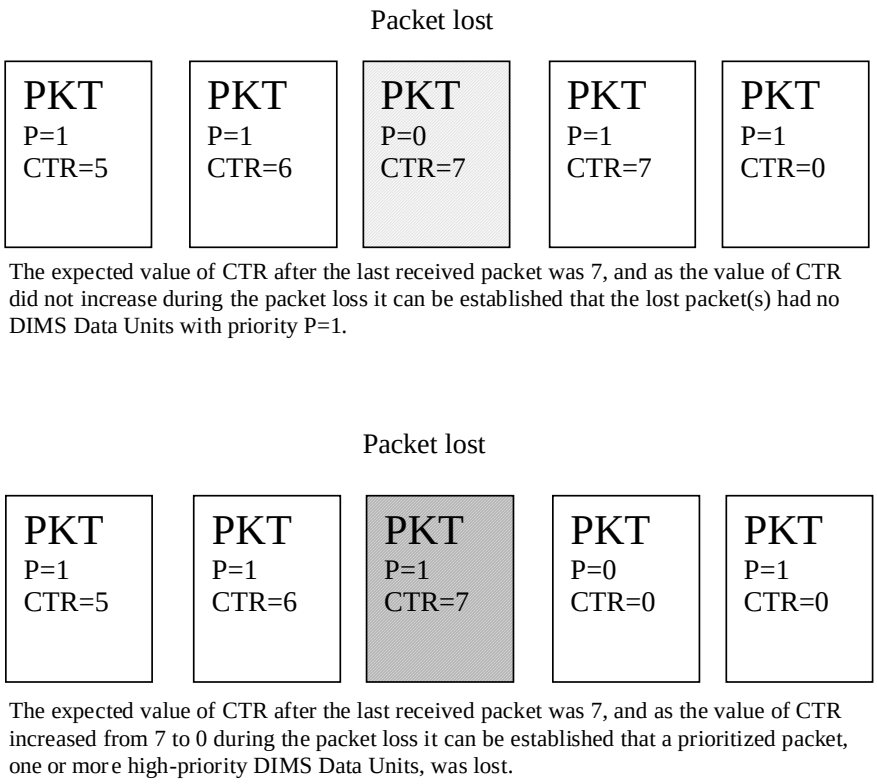


Figure 7-1: Example of prioritization including detection of lost prioritized packets

Note that loss is only detected on the next packet to arrive; if the content has long periods in which no packets are sent, or is otherwise bursty, it may be inadvisable to have a high-priority packet before a long silence interval, as its loss cannot be detected until the first packet after that interval, at the earliest.

7.3.2 RTP Packet format

7.3.2.1 Introduction

In the context of the present document (specifically the MIME type defined in subclause 11.1), the units carried by the RTP Payload Format are DIMS Units. The RTP payload format defines two basic packet structures:

a) packets containing one or more entire units;

b) packets containing a single fragment of a unit.

Depending on the underlying network and the unit size, it may be desirable to split units or aggregate them.

7.3.2.2 RTP Header Usage

The RTP header is defined in [15] and its use in this payload format is described below.

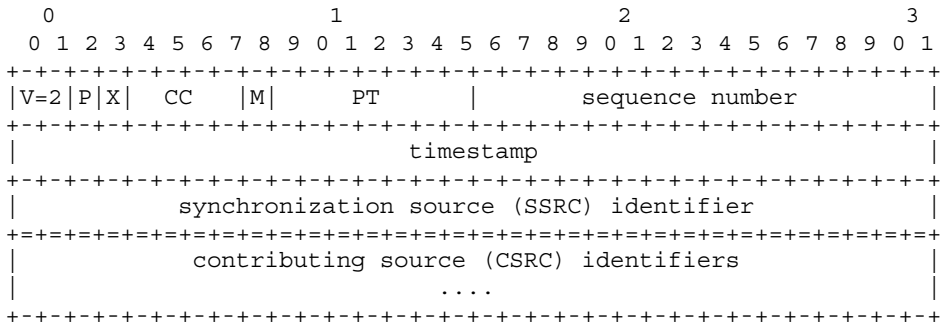


Figure 7-2: RTP HEADER

Marker bit (M): 1 bit - The marker bit is set for the last packet associated with a timestamp.

NOTE: This is useful when a scene is sent as a combination of a smaller scene and a series of scene commands in separate packets. In this case the marker bit of the packet containing the last scene command is to be set. This is in line with the normal use of the marker bit in video coding and enables efficient buffering.

Timestamp: 32 bits - The timestamp indicates the rendering instant of the unit(s).

The usage of the remaining RTP header fields follows the rules of [15].

7.3.2.3 Common Packet Header

The RTP payload comprises of a common header and has the following format:

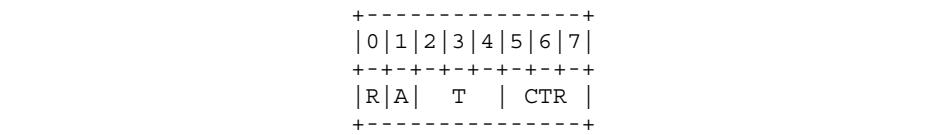


Figure 7-3: COMMON PAYLOAD HEADER

R: 1 bit

The R bit is reserved, shall be set to 0, and shall be ignored by the receiver.

A: 1 bit

When set to one, the A bit indicates that the packet contains one or more random access points (in DIMS, DIMS Units with is-RAP set), or the first fragment of a random access point.

T: 3 bits

The payload type as defined in table 2; Reserved values shall not be used, and packets with reserved values of the type field shall be discarded and not processed.

Table 2: Summary of RTP Payload Types and Descriptions

Type	Description
0	Aggregation packet
1	Fragmentation start Packet
2	Fragmentation continuing Packet
3	Fragmentation end Packet
4 to 7	Reserved

CTR: 3 bits

The CTR is used to detect the loss of one or more high-priority units as documented in subclause 7.3.1.

7.3.2.4 Aggregation Packet

These packets contain one or more complete units with the same timestamp. The common header values are:

- Type: 0.
- A (RAP): as needed.

The RTP payload is presented below.

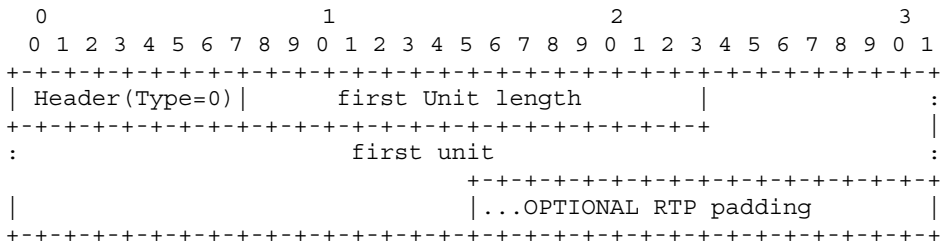


Figure 7-4: Aggregation Packet payload format

The units are placed in the RTP payload, in sequence, possibly following by RTP padding. Each unit is preceded by a two-byte length in network (big-endian) byte order. The length is the length of the following unit (both header and body), not including the length field itself.

7.3.2.5 Fragmentation Packets

Frames that exceed the networks maximum transmission unit (MTU) need to be fragmented before transmission. By fragmenting at the RTP level one need not rely on lower layer fragmentation, e.g. IP.

The payload format defines fragmentation of units into two or more RTP packets.

NOTE: Fragmentation on the RTP level should however be seen as a solution only when fragmentation on the DIMS level is not possible. Fragmentation can be performed by splitting, for example, a scene into a scene and a number of scene updates. In this way packets can be created that are smaller than MTUs and can be decoded individually, which gives better error resilience when packets are lost.

The common header values are as follows.

- Type: 1, 2, or 3.
- A (RAP): as needed in first fragment, and 0 in all other fragments.
- CTR: shall be identical in all the packets of a fragmented unit; increments after the last fragment depending on the priority of the unit.

Fragments consist of an integer number of consecutive octets of a unit. Fragments of a unit shall be sent as a group and in consecutive order with respect to RTP sequence numbers. The first fragment shall be marked as type 1 and the last fragment shall be marked as type 3. Other fragments shall be marked as type 2.

The unit is complete with header. The header is not repeated in fragments. There is no length field.

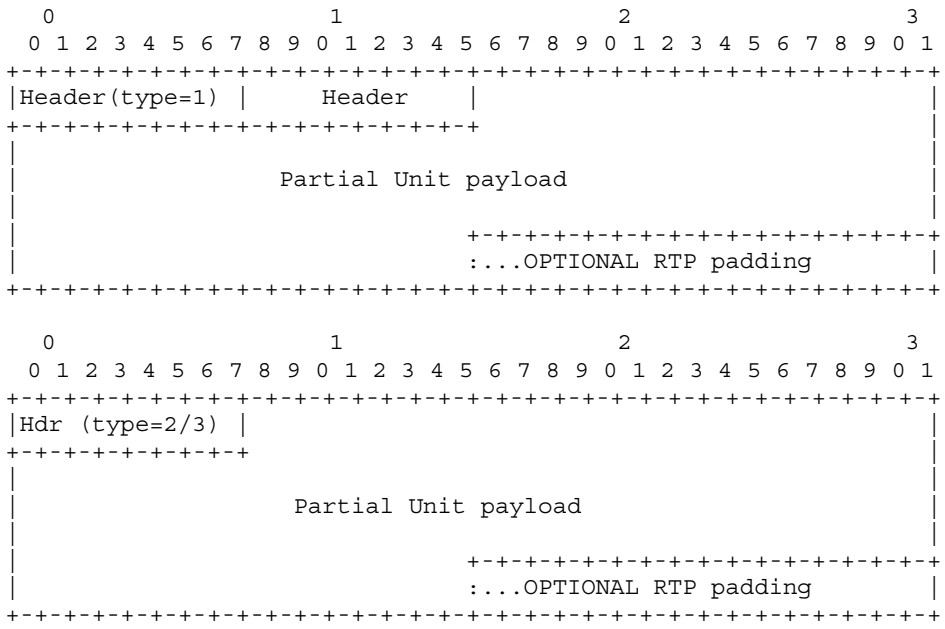


Figure 7-5: FRAGMENTATION PACKET FORMATS

7.3.3 SDP Parameters

The Session Description specifies the clock rate, version profile and level. The fields in the Session Description Protocol (SDP) are defined as follows:

- The media name in the "m=" line of SDP shall be video.
- The encoding name in the "a=rtpmap" line of SDP shall be richmedia+xml.

The clock rate in the "a=rtpmap" line is not specified in the present document. The resolution of the clock should be sufficient for the desired synchronization accuracy and for measuring packet arrival jitter. The clock rate of the referenced continuous media files within the presentation needs to be considered. For example, if the presentation contains referenced video which is to be synchronized with the presentation, the clock rate should be no less than 90,000.

The MIME parameters in subclause 11.1, when present, shall 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.

An example of a media-level description in SDP format is shown below.

```
m=video 12345 RTP/AVP 96
a=rtpmap:96 richmedia+xml/100000
a=fmtp:96 Version-profile=10; Level=20;
```

7.3.4 Separate Redundant Stream

Redundant DIMS Units may be carried in RTP in a separate stream. If there is more than one main stream, the redundant stream(s) shall be linked to the matching main stream(s) that they repair, using a suitable mechanism such as the media identification and group attributes as specified in [22].

Redundant streams have contains-redundant set to "redundant data only". The stream they are connected to shall have contains-redundant set to "main data only".

A terminal may perform tune-in etc. using the 'redundant' stream by:

a) looking for a random access point in the redundant stream, or main stream;

b) if the random access point was in the redundant stream, processing DIMS Units from the redundant stream until the redundant-exit indication;

c) continuing processing at the temporally next DIMS Unit in the main stream.

8 Profiles and Levels

8.1 Profiles

8.1.1 Introduction

A profile indicator in a stream indicates which features (also known as tools) are required to be supported on a terminal.

Profile indications are unsigned 8-bit integers. Only one profile is defined by the present document; other profiles may be defined in future.

When opening a bitstream, a terminal should check that the profile indicator is recognized by seeing whether the bitstream profile is equal to a profile supported in the terminal. Profile indicators can only be compared for equality; a terminal presented with an unknown profile indicator should assume it does not support the features required by the indicated profile.

The value 255 for the profile indication is defined to mean "no profile specified" and shall be used to mark bitstreams which are not known to conform to the requirements of any other profile.

The behaviour of a terminal when presented with a stream with an unrecognized profile, or profile 255, is unspecified.

8.1.2 Mobile profile

Mobile Profile : Profile Indicator Value 10.

Support for the following media types is also required in profile 10:

- Shall support the Still Image media format as specified in 7.5 of [7] and the Bitmap Graphics format as specified in 7.6 of [7].
- Shall support Speech as specified in 7.2 of [7] and Audio as specified in 7.3 of [7].
- Shall support Video as specified in 7.4 of [7].
- Shall support the 3GP file format as specified in 7.10 of [7].

As required in the SVG specification, SVG fonts shall be supported. The lack of hinting in SVG fonts means that small text which is anti-aliased may become unreadable. This problem is even more evident when text is rotated or animated. Recommendation: SVG fonts should be used with care.

The Open Font Format [4] should be supported at advanced simple text profile, level 2, with the following constraints: if Open Type fonts are supported, the DIMS client shall support downloadable OpenType fonts with TrueType outlines, TrueType hinting shall be supported for improved text readability, and advanced typographic features may be supported.

NOTE 1: When OpenType fonts are supported, download of them may be initiated using the font-face-uri element from [1].

Device-native fonts and fonts identified by generic family names may be used.

Uncompressed XML shall be supported. XML compressed with GZIP [11] shall be supported.

NOTE 2: Sub-systems may also require support for other media types (e.g. video) or codecs within those types (e.g. H.263) when support for DIMS is required.

8.2 Levels

8.2.1 Introduction

Level indicators provide a way to measure the degree of support required in a terminal to render a given scene or scene stream satisfactorily.

Level indications are unsigned 8-bit integers.

A level sets limits for quantitative parameters (e.g. a rotation angle) or measures (e.g. the overall bit-rate) of the format.

Level indicators are within the context of the profile indicator. All levels for a given profile use the same axes of measurement, but different profiles may use different axes; for example, one profile may have limits on rotation angle, and another profile leave that parameter un-limited. Within a given profile, the levels "nest", that is, if X<Y, then the requirements for a level with indicator X are lower than or equal to the requirements for a level with indicator Y, for the same profile.

The value 255 for the level indication is defined in all profiles to mean "no level specified" and shall be used in bitstreams not known to respect the limits of any known level for the indicated profile, and for all bitstreams labelled as "no profile specified". The behaviour of a terminal when presented with a stream with level 255 is unspecified.

DIMS Implementations will be able to use the level indicator to optimize the rendering of the content.

8.2.2 Level Axes

Levels may be measured on the following axes, or other parameters or measurable aspects of the stream. These values represent the maximum that the content may use, or expect., and hence the minimum the terminal must supply. For example, content must be usable if rendered into a minimum size pixel buffer and updated at the minimum pixel update rate.

1) Bitrate of the scene stream(s), including the initial scene, embedded graphics, audio, video, etc. (That is, the minimum bit-rate channel over which the scene could be delivered in a real-time fashion).

2) Bitrate of the stream(s) having the DIMS mediatype. (That is, excluding streams with other mediatypes, such as video, audio, etc.).

3) The maximum number of simultaneously playing video streams.

4) The maximum number of simultaneously playing audio streams.

5) The maximum number of simultaneously active DIMS Scenes.

6) The minimum output pixel buffer size needed for the scene to be usable. This is specified in nominally square pixels, as a width and height value (therefore 160x120 is wider than it is tall).

7) The maximum number of path segments over the entire scene.

8) The maximum total number of bytes in text content in text elements in the entire scene.

9) The maximum size of a single dash array.

10) The number of gradient stops in any single gradient.

The following limits should be respected by the content and should be supported by the terminal. However, these memory and CPU limits are for a reasonable scene with reasonable attribute values, in a scene which is un-scripted or has a reasonable amount of scripting. Use of large attribute values (e.g. long paths or strings), complex nodes, or scripts that are demanding in memory or processor usage will have an impact on performance and decodability.

1) The size of the DOM tree as measured by the number of nodes in the tree; the number of attributes, or the size of their values, is not calculated.

2) The expected minimum refresh rate of the pixel output buffer (e.g. as a result of animations, updates, or script actions) for the scene to be usable. This is expressed as a frame rate.

3) The maximum number of animations that run concurrently.

The following subclauses define the available levels.

8.2.3 Mobile Profile Level 10 definition

This level contains the following restrictions:

- Only one instantiation of a DIMS Scene is allowed.
- Only one Video instantiation along with a DIMS Scene is allowed.

On the Video Element, the attribute **transformBehavior** shall be restricted to values "pinned | pinned90 | pinned180 | pinned270", and the attribute **overlay** shall be restricted to values "top".

The following limits also apply:

Mobile Profile Level Limits

	Level 10
1 Scene bitrate (includes the static media embedded within the scene/commands and referenced media payloads)	200 kbit/sec
2 DIMS aggregate bit-rate (excludes media payloads)	40 kbit/sec
3 Simultaneous video playing	1
4 Simultaneous audio playing	1
5 Simultaneous active DIMS scenes	1
6 Minimum pixel output buffer size	160 wide by 120 tall
7 Maximum path segments across all paths	3000
8 Maximum text content size (this refers to the number of characters and does not include the glyph)	10k bytes
9 Maximum dash array size	16
10 Maximum number of gradient stops	32
1 Maximum recommended DOM tree size	800 nodes
2 Minimum recommended screen refresh rate	10 frames/sec
Note: Depending on the DOM tree size, the recommended screen refresh rate value may not be feasible and a lower screen refresh rate might be expected.	

3 Maximum simultaneous animations 20

8.2.4 Void

9 Content usage guidelines

The SVGT1.2 guidelines defined in ANNEX L of PSS release 7 [7] apply here.

NOTE: The recommendation of L.2.8 should be read as including downloadable fonts: "Usage of device or system fonts, or downloaded OpenType fonts, is recommended. SVG fonts should be used with care."

Content creators should take into account the DIMS levels definitions in section 8.2. Those level definitions do not completely assure interoperability in terms of memory or processing requirements. Content authors should be aware that CPU-intensive scripts, for example, may adversely impact update rates, or that heavily recursive scripts, or variables that hold large values, may adversely affect memory usage and hence player performance.

10 Security and Content Protection Considerations

DIMS does not define a security framework. DIMS relies instead on the security frameworks already defined for the mechanisms DIMS uses (e.g. for ECMAScript Mobile Profile [2]), and the frameworks provided by the platforms on which DIMS runs.

When content requests fullscreen video and especially fullscreen scenes, it is possible for the content to mimic the normal look of the device (the 'desktop' of a computer screen, for example) and persuade the user to enter potentially secure or private information into a presentation while thinking that they are interacting with the local system. This is sometimes called "phishing". Care should be taken to handle content that uses fullscreen requests, such that the user is always aware of when DIMS content is filling the screen (e.g. restrict DIMS to "window-only" mode, or in some other way make it clear to the user that the screen's content is rendered under DIMS control).

DIMS content can embed scripts. Care should be taken to limit, to the presentation in which they occur, the access that these scripts have. For example, it would normally be inappropriate for these scripts to have access to the local file system outside the scope of presentation. A further possible countermeasure would be to restrict DIMS reception to certified servers only, and to signed contents only, without any means for the user to disable this strict checking in order to mitigate social engineering attacks.

Authors of web-sites that embed DIMS content, when the scripts in the DIMS content are not under the control of the web site - for example, if the DIMS content is fetched from another site, or uploaded to the web-site by users -- should exercise caution. The embedded scripts may have access to the content of, and interaction of, the web site that embeds them, even though they were not authored by, or provided from, that web site.

Finally, it is recommended a secure client software update service be provided, so that it is possible to close security holes in the DIMS clients when they are detected

11 Registered Types

11.1 RTP Payload format MIME Type

Type name: video

Subtype name: richmedia+xml

Required parameters:

- `Version-profile` - A value between 0 and 255 specifying the profile of DIMS used, for example the value indicating Mobile Profile.

- `Level` - A value between 0 and 255 specifying the minimum DIMS level needed to be able to display the scene.

Optional parameters:

- `stream-type` - takes the value "primary" for primary streams (in which every random access point is a scene), and the value "secondary" for secondary streams. Secondary streams are not normally playable by themselves, outside the context of the scene(s) they are designed to update. The default value is "primary".

- `contains-redundant` - takes the value "main" if the stream contains only DIMS Units with is-redundant set to 0, the value "redundant" if the stream contains only DIMS Units with is-redundant set to 1, and takes the value "main+redundant" if both occur. Note that streams containing only redundant units must be linked to the matching stream carrying main DIMS Units (see subclause 7.3.4). The default value is "main+redundant".

- `text-encoding` - is a string enclosed in double-quotes with possible values taken the XML specification for character encoding in entities (e.g. subclause 4.3.3 of [25]). It describes the text encoding after the content has been decompressed (e.g. after deflating). The default value is "UTF-8" [9]. This field is only applicable if the content is transmitted as (possibly encoded) text.

- `content-script-types` - is a string enclosed in double-quotes that identifies the scripting languages used. It is formatted as a comma-separated list of MIME types [18] from the IANA registry, such as "application/ecmascript" (see [13]). It shall provide a complete listing of the script types that the terminal must support in order to process the stream. If this attribute is not present, the set of required script types is unknown. If this attribute contains the empty string, then the stream does not require any script processing.

- `content-coding` - this field provides the identification of the compression scheme. It is a string specifying the encoding (compression) format of the content. It is defined in the same way as the content-coding header in HTTP (subclause 3.5 of [12]). The default value is the empty string.

- `useFullRequestHost` takes the value "0" or "1"; the definition of this parameter is in subclause 5.5.2. The default value is "1".

- `pathComponents` takes a value between "0" and "15"; the definition of this parameter is in subclause 5.5.2. The default value is "15".

Encoding considerations:

- This media type is currently only defined for transport via RTP.

Security considerations:

- RTP packets using the payload format defined in the present document are subject to the security considerations discussed in the RTP specification [15] and any applicable RTP profile, e.g., AVP [21].

Interoperability considerations:

- None.

Published specification:

- 3GPP TS 26.142.

Applications that use this media type:

- DIMS Streaming applications.

Additional information:

- Magic number(s): None.

- File extension(s): None.

– Macintosh file type code(s): None.

Person and email address to contact for further information:

– Clinton Priddle.

– clinton.priddle@ericsson.com.

– Multimedia Technologies, Ericsson.

Intended usage:

– COMMON.

Restrictions on usage:

– None.

Author:

– 3GPP SA4 WG.

Change controller:

– 3GPP TSG SA.

11.2 'Codecs' Parameter for 3GP files

When DIMS content is supplied in 3GP files which are identified by MIME type, the 'codecs' parameter defined in [14] may be used to indicate that DIMS content is present. The codecs parameter takes the sample entry name as defined above (that is, 'dims').

Annex A (normative): Conformance Criteria

DIMS constructs scenes which are possibly updated over time. Conformant terminals shall support the delivery of the scenes and updates in formats specified in the 'compression' subclause above, and in the transport environments specified in the 'transport' subclause.

For the initial scene, a DIMS Scene can be extracted from the transport, and de-compressed if necessary, yielding an XML document. This XML is referred to here as the "initial DIMS document". Similarly, after all updates for a given instant have been applied to the scene tree, there is logically an XML document that is equivalent to the scene DOM tree; these are called "subsequent DIMS documents" here.

Initial and subsequent DIMS documents shall conform to all of:

- the conformance requirements in Appendix D of [1]; with the following exceptions:
  - The conformance criteria in the SVG specification regarding codecs do not apply for the DIMS media type.
  - Clause D.4 is not in scope for DIMS.
  - Clause D.7 is not in scope for DIMS.
- the conformance requirements of the LASeR Commands and LASeR scene extensions as specified in [27] and [28]
- the limitations of the profile and level indications under which they are delivered.

Conformance of the DIMS extensions is for further study.

Annex B (informative): Change history

Change history							
Date	**TSG #**	**TSG Doc.**	**CR**	**Rev**	**Subject/Comment**	**Old**	**New**
2007-06	36	SP-070312			Approved at SA#36 Plenary		7.0.0
2007-09	37	SP-070632	0006	1	Technical corrections in the DIMS specification	7.0.0	7.1.0
2007-12	38	SP-070764	0007		Reference correction in the DIMS specification	7.1.0	7.2.0
2009-06	44	SP-090247	0009	1	DIMS Mobile profile level 10 review and clarification	7.2.0	7.3.0

History

Document history		
V7.0.0	June 2007	Publication
V7.1.0	October 2007	Publication
V7.2.0	January 2008	Publication
V7.3.0	June 2009	Publication