Rich Media Environment Technical Specification

Approved Version 1.0 – 29 Mar 2011

Open Mobile Alliance

OMA-TS-RME-V1_0-20110329-A

Contents

Tables

1. Scope

The Scope of this document is to specify all the components, interfaces and functionalities needed by the Rich-Media Environment Enabler, according to the [RME_RD] and the [RME_AD] documents.

A joint work has been agreed to define the RME enabler with the 3GPP SA4 working group. As a consequence the RME TS is based on the 3GPP SA4 Dynamic and Interactive Multimedia Scene [DIMS]

The objectives of this document, detailed below, are:

- To complement [DIMS] in OMA specific areas
- To ensure compliancy and compatibility with [DIMS]
- To fulfill RME Requirements not taken in account by [DIMS]
- To ensure RME will remain bearer agnostic and usable outside of 3GPP networks

Specific OMA areas

In addition to the RME requirements not fulfilled by [DIMS] or fulfilled in a 3GPP specific manner, the RME specification needs to complement [DIMS] in areas owned by OMA such as:

- Specific functions (e.g. Soft key mapping, event handling and fonts)
- Overall Processing Model
- UAProf vocabulary
- Integration with other OMA enablers
- Alignment with related mark-up languages (e.g. XHTML).

Compliance with DIMS

A coordination process has been pursued with the 3GPP SA4 working group since the delivery of the OMA RD document. The intent between the BT-MAE and 3GPP SA4 is to develop a common Core Rich-Media specification usable on any bearer, but well adapted to 3GPP networks. In order to avoid market fragmentation, the RME TS will reference the [DIMS] specification in sections related to the core common rich-media specification and will specify additional features when needed for the OMA environment. The RME TS is defined as a superset of [DIMS].

DIMS and RME share the same MIME type definition: video/richmedia+xml for stream-based RTP transport of rich media data.

RME requirements not taken in account by DIMS

When developing DIMS, the 3GPP SA4 working group has delivered the document [DIMS REQ] which adapted the RME requirements to DIMS needs. In particular Annex A of [DIMS REQ] lists all the RME requirements not taken in account for the [DIMS] specification development. The RME specification should define the complementary text for these requirements.

Usage of RME outside of 3GPP networks/environment

The RME requirement document states that the RME enabler is agnostic of any bearers, networks, OSs, etc. [DIMS] is not totally agnostic of “any” bearers outside of 3GPP networks and environment.

RME intends to maximize the usage of common technologies across various environments where possible.

The RME specification needs to provide equivalent functionalities to [DIMS] for non 3GPP environment, e.g.: 3GPP2, DVB, OMA-PUSH.

2. References

2.1 Normative References

[BCAST] "Mobile Broadcast Services", Version 1.0 – 7 August 2008, OMA-TS-BCAST_Services-V1_0-20080807-C, URL: http://www.openmobilealliance.org/

[DCD] "Enabler Release Definition for DCD", Version 1.0 – 23 June 2008, OMA-ERELD-DCD-V1_0-20080623-D, URL: http://www.openmobilealliance.org/

[DIMS] "Dynamic Interactive Multimedia Scenes (DIMS)", version 7.1.0, 3GPP specification, URL: http://www.3gpp.org/ftp/Specs/html-info/26142.htm

[DOM3_EVENT] "Document Object Model (DOM) Level 3 Events Specification", World Wide Web Consortium, Working Draft, 21 December 2007, URL: http://www.w3.org/TR/DOM-Level-3-Events

[ESMP] "ECMAScript Mobile Profile", Version 1.1, Open Mobile Alliance™, OMA-TS-ESMP-V1_1, URL: http://www.openmobilealliance.org/

[LASER] 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

[MAXE] "Media Access Events", World Wide Web Consortium Working Draft, 13 October 2006, URL: http://www.w3.org/TR/MediaAccessEvents/

[OFF] ISO/IEC 14496-22 *Information technology — Coding of audio-visual objects — Part 22: Open Font Format*

[PUSH] "Enabler Release Definition for Push", Version 2.2, Open Mobile Alliance™, OMA-ERELD-Push-V2_2-200710002-C, URL: http://www.openmobilealliance.org/

[RFC2046] "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types" N. Freed and N. Borenstein, November 1996, URL:http://www.ietf.org/rfc/rfc2046.txt

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997, URL: http://www.ietf.org/rfc/rfc2119.txt

[RFC2387] "The MIME Multipart/Related Content-type", E. Levinson, August 1998, URL: http://www.ietf.org/rfc/rfc2387.txt

[RME_AD] "Rich Media Environment Architecture", Open Mobile Alliance™ OMA-AD-Rich_Media_Environment-V1_0, URL: http://www.openmobilealliance.org/ftp/Public_documents/BT/MAE/Permanent_documents/OMA-AD-RME-V1_0_0-20070615-D.zip

[RME_RD] "Rich Media Environment Requirements", Open Mobile Alliance™, OMA-RD_Rich-Media-Environment-V1_0, URL: http://member.openmobilealliance.org/ftp/Public_documents/BT/MAE/Permanent_documents/OMA-RD-Rich-Media-Environment-V1_0-20050923-C.zip

[SCRRULES] "SCR Rules and Procedures", Open Mobile Alliance™, OMA-ORG-SCR_Rules_and_Procedures, URL: http://www.openmobilealliance.org/

[SMIL] "Synchronized Multimedia Integration Language (SMIL 2.1)", World Wide Web Consortium, Recommendation, URL: http://www.w3.org/TR/2005/REC-SMIL2-20051213/

[SVG] "Scalable Vector Graphics (SVG) Tiny 1.2 Specification", World Wide Web Consortium Recommendation, URL: http://www.w3.org/TR/SVGMobile12/

[UAPROF] "User Agent Profile", , Version 2.0, Open Mobile Alliance™, OMA-TS-UAProf-V2_0 URL: http://www.openmobilealliance.org/

[URISCH] "URI Schemes for the Mobile Applications Environment", Version 1.0, Open Mobile Alliance™, OMA-TS-URI_schemes_V1_0 URL: www.openmobilealliance.org/

2.2 Informative References

[DIMS REQ] "DIMS Technical requirements", 3GPP document S4-050800,
URL: http://www.3gpp.org/ftp/tsg_sa/WG4_CODEC/TSGS4_37/Docs/S4-050800.zip

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

3.2 Definitions

Normal Play Time	The normal play time at a particular instant in a stream is equal to the time it takes to play the stream, at normal pace and without interrupts, from the start of the stream to the particular instant.
RME data	XML data consisting of either initial scene data or scene commands.
RME timeline	Timeline graded by the normal play time of an RME stream.
RME Unit	The smallest process-able unit of RME data. For streaming delivery the RME Unit is equivalent to DIMS Unit as defined in Section 5.6 [DIMS]. For HTTP delivering the RME Unit equals the body of the DIMS unit.
Rich media	Media consisting of aggregation of a number of different media types such as vector graphics, still images, video and audio.
Scene	The description of temporal and spatial layout of objects (included or by references) such as vector graphics, images, audios and their linkage to each other
Scene Commands	Information defining modifications to a scene.

3.3 Abbreviations

3GPP	Third Generation Partnership Program
AAC	Advanced Audio Coding
AD	Architecture Document
API	Application Programmer’s Interface
BCAST	OMA Mobile Broadcast Services Enabler Suite
DCD	Dynamic Content Delivery enabler
DIMS	Dynamic and Interactive Multimedia Scenes
DOM	Document Object Model
DRAP	Dynamic Random Access Point
DRM	Digital Rights Management
DVB	Digital Video Broadcasting
ECMA	European Computer Manufacturers Association
ESMP	ECMA Script Mobile Profile
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
MBMS	Multimedia Broadcast Multicast Service
OMA	Open Mobile Alliance
RD	Requirement Document

RME	Rich Media Environment
RTP	Real-time Transport Protocol
RU	RME Unit
SDP	Session Description Protocol
SVG	Scalable Vector Graphics
TS	Technical Specification
uDOM	Micro DOM
URL	Uniform Resource Locator
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language

4. Introduction

The rich media environment (RME) specification defines a language for describing rich media content, which can be distributed to and displayed on mobile devices. RME content consists of scenes of visual objects, such as video, images, animation and text, and audio objects that are composed together to give the user a richer experience. The system contains means to continuously update the scene with new information replacing only the parts that are changing. It will thus be possible to keep part of the scene while updating other parts of the scene, thereby saving both communication bandwidth and device processing power. It is however, also possible to replace the current scene with a new scene, if desired.

Typical applications for RME are mobile TV Clients, dynamic application user interfaces, multi-player gaming, and on-device portals.

The RME system consists of the RME Client and the RME server. The RME Client typically resides on the mobile terminal and provides the capability to display RME data, handle dynamic updates to the RME Scene as well as local and remote interaction with the scene objects. Typically the server is the source of data and provides RME data to the client.

5. Namespace definition

The RME specification is an extension of the DIMS specification defined in [DIMS]. All XML entities defined in DIMS are in the namespace “http://www.3gpp.org/richmedia/”. The current specification defines additional XML entities.

Although the development of the two specifications has been done in close cooperation, a new namespace is defined here to avoid the risk of name clashing, and in addition, to serve as a placeholder for functional entities defined in this document. XML entities defined in the current document are defined in the namespace “http://www.openmobilealliance.org/richmedia”.

6. Scene Description

This section defines the XML language constructs that are used to describe an RME scene. The scene description defines visual and audio objects and the layout of these objects on the drawing canvas. The description of a scene also defines any temporal relation between the objects in the scene, as well as local and remote interaction with scene objects.

6.1 SVG Tiny 1.2

The scene description language is based on the SVG Tiny 1.2 specification [SVG]. The RME scene description language SHALL support all language features that are included in the SVG feature string http://www.w3.org/Graphics/SVG/feature/1.2/#SVG-all.

6.2 Scene Description Extensions

The scene description language in addition to the SVG Tiny 1.2 as defined in Section 6.1, SHALL support the extensions defined in Section 5.4.2 in [DIMS]. These extensions comply with SVG Tiny 1.2 Extension Conformance Requirements in Section D.6 in [SVG].

6.3 URI support

The RME enabler SHOULD support URI schemes according to [URISCH].

7. Scene Commands

This section defines XML language constructs used to accomplish changes to an RME scene. The commands can be synchronized to the time line of the scene.

7.1 Update Commands

The update commands are used to modify an RME scene. The update commands allow insert, delete, replace and add operations on the RME scene. The syntax and semantics of the update commands are defined in Section 5.5.1 in [DIMS] and SHALL be supported by the RME enabler. The update commands are based on the MPEG4 part 20 specification [LASER].

7.2 State Management Commands

The RME enabler provides state management capabilities through the save/restore/clean commands, which operate on DOM attributes. The mechanisms are specified in Section 5.5.2, in [DIMS] and SHALL be supported by the RME enabler.

The state management mechanisms can be a potential security risk. The security measures defined in 5.5.2 of [DIMS] SHALL be used.

An RME enabler implementation may impose additional security mechanisms.

7.3 Activation Commands

The RME system SHALL support the activate/deactivate commands as specified in Section 5.5.3 in [DIMS]. These commands operate on the nodes of the DOM tree and provide means to temporarily activate/deactivate a branch of the DOM tree.

7.4 Script as a Command

Scripts can be sent to the client for immediate execution. Scripts SHALL have access to the RME Scene, including the ability to modify the DOM. The RME Client SHALL NOT load these scripts in the DOM. The command for sending such scripts is defined in Section 5.5.5 in [DIMS] and SHALL be supported by the RME enabler.

7.5 Seeking in the RME timeline

To perform relative seek backward and forward in the RME timeline, the seek command as defined in Section 5.5.6 in [DIMS] SHALL be supported by the RME enabler for locally stored data (for e.g. RME content packaged in a 3gp file). The device is NOT REQUIRED to buffer any data from a data stream. The RME enabler MAY support this command for streamed data.

7.6 Scene Command document

The `<sceneCommandGroup>` element defines a wrapper element for scene commands. The `<sceneCommandGroup>` element is defined in the RME namespace and has no attributes. By wrapping scene commands in this element a valid XML document is created.

Download requests for `<sceneCommandGroup>` documents can be made from the scene using the <updates> element as defined in Section 5.4.2.6 in [DIMS] or scripts. The commands within a <sceneCommandGroup> SHALL be applied in the order they appear in the element.

The retrieved commands do not have any media time and SHALL be applied according to the begin time of the <updates> element for update requests and as soon as they arrive for script requests.

8. Scripting functionality

This section defines the scripting capabilities available in the RME enabler. The RME enabler SHALL include support for script execution. Scripts are loaded to the client either as part of the initial scene (see Section 5), as an update (see Section 7.1) or in an immediate script execution unit (see Section 7.4).

Scripts used in RME context may pose security risks. RME does not define any security mechanisms of its own for script execution and relies on the protection mechanisms provided by the underlying scripting environments. The authors and implementations are recommended to exercise caution when scripts are used in the context of RME.

8.1 Scripting language

The RME Client SHALL support the ECMA Script Mobile Profile as defined in [ESMP] and MAY support other scripting languages.

8.2 Script processing

The scripts supplied with the initial scene or in scene updates are loaded into the DOM of the scene and SHALL be executed according to the rules in Section in 15.2.1 (SVG Script Processing) in [SVG].

Scripts sent from the server to the client according to Section 7.4 SHALL be executed according to Section 10.1in the present specification. These scripts SHALL NOT be loaded into the DOM and SHALL be removed as soon as they have finished their execution.

8.3 uDOM API

The RME system SHALL allow scripts to access the RME Scene using the uDOM API as defined in appendix A in [SVG].

8.4 Extensions to uDOM API

The RME system SHALL in addition to the uDOM API defined in Section 8.3 support the APIs necessary for scene extensions as defined in Section 6.3 in [DIMS].

8.5 ESMP bindings to uDOM

The uDOM used in RME SHALL bind to the ECMA script mobile profile using the bindings defined in Appendix O in [SVG]. The extensions to the uDOM defined in this specification SHALL be bound to ECMA script mobile profile using the bindings defined in Appendix D in the present specification.

9. Events

The RME enabler supports event creation, dispatching and handling on the client. An RME Client SHALL handle events according to the DOM-Level-3 Events specification as specified in Section 13 in [SVG].

9.1 List of Supported Events

The RME Client SHALL support creation and dispatching of the following events:

- All events of SVG Tiny 1.2 as defined in Section 13.2 in [SVG]
- The Media Access Events as defined in [MAXE]
- Events defined in the DIMS specification Section 6.1 in [DIMS]
- Events defined in Section 9.2 in the present specification

The Event object property "type" SHALL map to "Event identifier", as defined by SVG, for all events defined outside of the SVG specification.

9.2 RME defined events

9.2.1 Screen Orientation event

The RME enabler SHALL support the screen orientation events as defined in Section 5.4.2.8 in [DIMS]. The RME enabler SHALL support the ScreenOrientationEvent interface as defined below. This interface is a superset and redefinition of the interface with the same name defined in Section 6.1.3 in [DIMS]. The superset is a result of the softKeysLocation attribute not present in DIMS.

This interface provides contextual information associated with change in Screen Orientation modes. Supported event types are **ScreenOrientationPortrait** and **ScreenOrientationLandscape.**

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

Attributes

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 un-rotated horizontal axis with a value between 0 and 359 inclusive.

softKeysLocation - 3-tuples of screen coordinate X, screen coordinate Y, and Key Identifiers.

9.2.2 Key Identifiers

The RME enabler SHALL, in addition to the key identifiers defined in [DOM3_EVENT] support the following key identifiers in the KeyboardEvent interface:

“Softkey_1” – The primary softkey.

“Softkey_2” – The secondary softkey.

“Softkey_3” – Optional softkey.

“Softkey_n” – Additional optional softkeys 4 and beyond where *n* indicates the number.

“Joystick_Left” – Indicates ‘left’ movement by the rocker/9-way switch.

“Joystick_Right” – Indicates ‘right’ movement by the rocker/9-way switch.

“Joystick_Up” – Indicates ‘up’ movement by the rocker/9-way switch.

“Joystick_Down” – Indicates ‘down’’ movement by the rocker/9-way switch.

“Joystick_Center” – Indicates ‘center’ movement by the rocker/9-way switch.

“Joystick_UpLeft” – Indicates ‘up/left’ movement by the rocker/9-way switch

“Joystick UpRight” - Indicates ‘up/right’ movement by the rocker/9-way switch

“Joystick DownLeft” - Indicates ‘down/left’ movement by the rocker/9-way switch

“Joystick DownRight” - Indicates ‘down/right’ movement by the rocker/9-way switch

9.3 Event Processing Model

The processing of events SHALL follow the DOM-Level-3 Events specification as used in Section 13 in [SVG].

The RME enabler MAY support the capture phase completion of the DOM Events flow.

Event processing SHOULD be considered an atomic action. After an event has been created and dispatched, the processing of the event SHOULD continue until completion. For further information see Section 1.2.2.5 in [DOM3_EVENT].

10. Timing Model and Processing Model

In this section the overall processing and timing model is defined. This is the relation between processing and timing of the RME scene objects: Scripts, events, updates and the regular scene objects. The intrinsic timing and processing details of the objects themselves is discussed in their respective section.

10.1 Timing model

The RME enabler processes the following objects: Initial scenes, scene commands, events, scripts and rendering. All processing objects SHALL be processed at their normal play time. Objects of different types but with the same play time can be processed in implementation specific order. Objects of the same type and with the same normal play time can be handled in any order.

Objects in the RME scene can be synchronized relative to each other. The synchronization requires that timing information is available for all elements that are to be synchronized. Obtaining timing information for the media elements is dependent on the particular format used for the underlying media of these elements.

Scene commands can be synchronized to the RME Scene time. The command structure does not contain timing information and relies on the transport layer timing information. The correlation between the transport layer timing and the normal play time is handled by the transport layer.

The definition of transport layer time in different delivery contexts is defined in Section 11 below.

10.2 Processing Model

The processing model for handling the RME units is defined in Section 5.8 in [DIMS].

The playback algorithm of a compliant RME Engine SHALL produce the same result as the algorithm described below with the following high-level steps for each execution cycle:

1. Load the initial RME scene and initialize the scene time;
2. Compute the new scene time *t* (begin of execution cycle);
3. Decode any RU with a scene time below or equal to *t*, and not yet presented in earlier execution cycles, as well as any RU with no specified time;
4. Execute commands from RUs decoded at step 3;
5. Event resolution and media synchronization
    5.1. Process all events according to the DOM event model [DOM3_EVENT] and resolve all begin and end times that can be resolved according to the SMIL Timing Model, in clause 10 of [SMIL];
    5.2. Determine active media objects by inspecting begin and end times,
    5.3. If there are more events generated, go to 5.1
6. For each active media object, calculate its normal play time, and prepare for rendering of next frame.
7. Render the audio and visual element of the scene tree according to the SVG rendering model as described in Clause 3 of [SVG] (end of execution cycle).

11. Packaging and delivery

The RME data can be packaged, accessed and delivered through the following mechanisms:

- RME document
- 3gp file
- Multipart MIME
- Streamed sequences of RME Units
- Local storage

The primary addressed delivery context in this specification is 3GPP networks. Additional delivery contexts can be added in future releases. Some delivery contexts e.g. broadcast streaming context, require the client to perform tune-in.

This section also defines how the media time for updates is obtained in different delivery contexts.

11.1 RME document

An RME document SHALL consist of either a scene (see Section 6) or scene commands packaged in a <sceneCommandGroup> element as defined in Section 7.6in the current specification. The document SHALL constitute a valid XML document.

The MIME type for Initial scenes SHALL be “application/richmedia+xml”.

The MIME type for <sceneCommandGroup> documents SHALL be “application/richmediacommand+xml”.

An RME documents containing a scene command groups is recommended to be stored with the file extension “scg” as a matter of best practice. [1]

It SHALL be possible to deliver the RME document using point to point (e.g.: HTTP), or broadcast/multicast (FLUTE) mechanisms.

11.1.1 Specific data signaling

RME data transported as documents does not contain the DIMS Unit header part. The table below indicates how the information in the DIMS unit header is retrieved when the RME data is transported as RME documents.

Field	Signaling in DIMS (RTP/3GP)	Signaling in RME (HTTP)
S	Indicates if the body is a scene or not	Signaled in the HTTP Content-Type (MIME) field (see Section 11.1)
M	Indicates if the body is a random access point	Not of interest in error free transport without tune-in
I	Signals redundant body	
D	Signals exit of multi unit random access point	
P	Signal if the unit is essential or not	
C	Signals if the body is compressed	Signaled in the HTTP Content-Encoding field

Table 1 DIMS unit header data and how it is retrieved for RME data in RME documents.

The security variables *useFullRequestHost* and *pathComponents* are in [DIMS] only defined for streamed delivery of data. For RME data delivered as RME documents the security variables *useFullRequestHost* and *pathComponents* shall be set as

[1] The mime type should be review if we decide to use the OMA process or to do an IETF registration.

name=value pairs as extensions to the MIME type in the Content-Type identification of the document. The usage of these parameters SHALL be supported as defined in Section 5.5.2 in [DIMS].

Example Usage:

```
Content-Type = application/richmedia+xml;useFullRequestHost=1;pathComponents=4
```

11.2 3gp file

The RME enablers SHALL support packaging of RME data into 3GP files according to Section 7.2 in [DIMS]

The 3GP file format supports progressive download. In this case the media time for the updates SHALL be the sample times in the 3GP file.

11.3 Multipart MIME

11.3.1 Multipart MIME RELATED

The RME enabler SHOULD support multipart MIME delivery of the type RELATED [RFC2387]. Multipart RELATED is a transport packaging mechanism allowing related MIME messages e.g. initial scene and updates to the scene, to be transmitted in one request.

Example usage of Multipart MIME:

```
---------Multipart Message - Begin--------------------------
Content-type: multipart/related; boundary="RME"

--RME
Content-type: application/richmedia+xml
<svg>
     <text>This is your initial scene</text>
</svg>
--RME
Content-type: application/richmediacommands+xml
<sceneCommandGroup>
      <insert>…...</insert>
      <add>……..</add>
      <delete>……</delete>
      <doScript>……</doScript>
</sceneCommandGroup>
--RME--
---------Multipart Message - End--------------------------
```

11.3.2 Multipart MIME MIXED

The RME enabler SHOULD support multipart MIME delivery of the type MIXED [RFC2046]. Multipart MIXED is a transport packaging mechanism allowing unrelated MIME messages e.g. updates, to be transmitted in one request.

11.4 Streams of RME units

A packet of streamable RME data SHALL consist of an RU defined as the DU in Section 5.6 in [DIMS]. The RME enabler SHALL support RTP delivery of RUs using the payload format and SDP parameters defined in Section 7.3 in [DIMS].

The streaming distribution can be of the type one-to-one (e.g.: PSS) and one-to-many (e.g.: BCAST).

The MIME type for streaming over RTP SHALL be "**video/richmedia+xml**".

In this case, the media time for updates SHALL be the RTP time stamps.

11.4.1 Random Access Point

Random access points are used to perform tune-in, random access and error recovery in a RTP streaming environment. The RME enabler SHALL support the Random Access Mechanism as defined in Section 5.9 in [DIMS] including support for Dynamic Random Access Points DRAP specified in Section 5.5.4 in [DIMS].

11.5 Storage of RME data locally

The RME Client SHALL support retrieving 3GP files and RME data including RME documents from the local file system of the device.

11.6 Other delivery contexts

Besides the delivery mechanism defined in Section 11.1 and 11.4.1 other delivery contexts can provide desirable functionality for an RME service. RME does not preclude any delivery contexts that usefully provide the client with RME data. Particularly the OMA-BCAST [BCAST] enabler, the OMA-PUSH [PUSH] enabler and the OMA-DCD [DCD] enabler MAY be used.

12. Capabilities, Configuration and Management

12.1 Static Capabilities

Static device capabilities for RME are advertised using the facilities provided by OMA UAProf [UAPROF]. The following UAProf vocabulary is defined, and SHOULD be supported by device profiles describing RME capable devices:

Attribute	Component	Description	Resolution	Datatype	Example
RMECapable	SoftwarePlatform	Indicates whether the device provides Rich Media User Agent Services.	Locked	Boolean	Yes, No
RMEVersion	SoftwarePlatform	Defines version of the Rich Media Enabler (RME) capability within the device.	Locked	Literal	"1.0"
RichMediaDeliveryContext	SoftwarePlatform	List of RME Delivery contexts the device supports for transporting RME data streams. Property value is a list of supported contexts, where each item in the list is a context name.	Append	Literal, Bag	"HTTP", "RTP", "FLUTE"
RichMediaPackagingFormats	SoftwarePlatform	List of packaging formats the device supports for packaging Rich Media data streams. Property value is a list of supported formats, where each item in the list is a packaging format name.	Append	Literal, Bag	"3GP", "multipart", "MULTIPART"
SupportedScreenOrientations	SoftwarePlatform	Describes the list of supported screen orientation capabilities. Default "landscape" is defined as 0 degrees. Default "portrait" is defined as 90 degrees. References: 3GPP 26.142 v7.1.0 Section 5.4.2.8 OMA RME TS Section 9.2.1 MPEG 20	Append	Integer, Bag	0,90,180, 270
ScreenDefaultOrientation	SoftwarePlatform	Describes the default orientation of a device as either 0 (landscape) or 90 (portrait).	Locked	Integer	90

12.2 Dynamic Capabilities

Dynamic capabilities are currently available via ECMAScript API interface (see Section 9.2.1). When dynamic capabilities reporting is implemented in the networked environment, remote access to this vocabulary MAY be provided.

The following vocabulary is defined, but, because of its dynamic nature need only be supported through the specified ECMAScript API:

Attribute	Component	Description	Resolution	Datatype	Example
ScreenOrientation	TBD DCAP	Shows the current screen orientation. ”portrait” is always long dimension vertical. “landscape” is always long dimension horizontal. An angular reference assumes “landscape” is 0 or 180 degrees and “portrait” is 90 or 270 degrees. (definition matches with ScreenOrientations)	Dynamic	Integer	0,90,180,270
RichMediaScreenHeight	TBD DCAP	Contains the current running screen viewport height for a DIMS/RME application in pixels. Height is relative to CurrentScreenOrientation as the origin is always located at the top-left corner of the screen.	Dynamic	Integer	500
RichMediaScreenWidth	TBD DCAP	Contains the current running screen viewport width for a DIMS/RME application in pixels. Width is relative to CurrentScreenOrientation as the origin is always located at the top-left corner of the screen.	Dynamic	Integer	300

13. Host Services

13.1 Font

RME Client implementations SHALL support downloadable OpenType fonts [OFF] with TrueType outlines. TrueType hinting SHALL be supported for improved text readability, and advanced typographic features MAY be supported.

SVG font support is included as part of the SVG specification.

See Appendix C for guidelines on font usage.

Downloadable OpenType fonts SHALL be identified in the RME content using ‘font-face-src’ and ‘font-face-uri’ elements from [SVG].

13.2 Video

The video formats that are made available by the client device SHALL be accessible by the RME Client.

13.3 Image

The image formats that are made available by the client device SHALL be accessible by the RME Client.

13.4 Audio

The audio formats that are made available by the client device SHALL be accessible by the RME Client.

Note: In the ETR the required codecs will be specified on a per delivery context basis (e.g.: H263 and AAC+ for MBMS)

13.5 Content security and protection

The RME enabler does not define a particular content protection or security mechanism. The RME enabler MAY use the OMA DRM enabler to provide content protection. The usage of OMA DRM is implementation dependent.

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-TS-RME-V1_0-20110329-A	29 Mar 2011	Status changed to Approved by TP: OMA-TP-2011-0096-INP_RME_V1_0_ERP_for_Final_Approval

Appendix B. Static Conformance Requirements (Normative)

The notation used in this appendix is specified in [SCRRULES].

B.1 SCR for RME Client

Item	Function	Reference	Requirement
RME-SCENE_DESC-C-001-M	Support for basic scene description	Section 6.1	SVG:MCF
RME-SCENE_DESC-C-002-M	Support for extended scene description	Section 6.2	
RME-SCENE_ COMMAND-C-001-M	Support for scene mutation commands	Section 7.1	
RME-SCENE_ COMMAND-C-002-M	Support for scene state management commands	Section 7.2	
RME-SCENE_ COMMAND-C-003-M	Support for scene activation command	Section 7.3	
RME-SCENE_ COMMAND-C-004-M	Support for script command	Section 7.4	
RME-SCENE_ COMMAND-C-005-M	Support for setting stream time	Section 7.5	
RME-SCENE_ COMMAND-C-006-M	Support for command grouping	Section 7.6	
RME-SCRIPTING-C-001-M	Support for script execution and integration	Section 8	ESMP-Language-C:MCF
RME-EVENT-C-001-M	Support for inherited events	Section 9.1	
RME-EVENT-C-002-M	Support for locally defined events	Section 9.2	
RME-EVENT-C-003-M	Support for event processing	Section 9.3	
RME-PROCESSING-C-001-M	Conform to RME processing rules	Section 10	
RME-DELIVERY-C-001-M	Support RME data delivered in RME documents	Section 11.1	
RME-DELIVERY-C-002-M	Support for RME in delivery contexts	Section 11	
RME-DELIVERY-C-003-M	Support RME data delivered in 3gp files	Section 11.2 Section 11.5	
RME-DELIVERY-C-004-O	Support Multi part MIME	Section 11.3	
RME-DELIVERY-C-005-M	Support RME data delivered in RTP streams	Section 11.4	
RME-RESOURCES-C-001-M	Support for resource usage	Section 13.1 Section 13.2 Section 13.3 Section 13.4	

Appendix C. Font Usage for Content Authoring Guidelines (Informative)

It is strongly recommended that the original (unmodified) OpenType / TrueType fonts be used when support for custom fonts is required. In order to reduce the size of downloaded font data, fonts may be subsetted to include only those glyphs that represent characters in use by RME content. Please note that the process of conversion of OpenType / TrueType fonts to SVG font format is lossy. The conversion of fonts may also be prohibited by a font license. Please also note that the use of SVG fonts may increase the size of downloaded font data, may limit the ability to support some languages and will affect the text legibility (see [SVG], subclause 17.2).

Appendix D. Definition of ECMA Script Bindings for uDOM Extensions

Object TraitAccess

getFloatTraitNS(namespaceURI, name)

This method returns a(n) Number. The **namespaceURI** parameter is of type **String**. The **name** parameter is of type **String**.

setFloatTraitNS(namespaceURI, name, value)

This method has no return value. The **namespaceURI** parameter is of type **String**. The **name** parameter is of type **String**. The **value** parameter is of type **Number**.

Object ScreenOrientationEvent

ScreenOrientationEvent has all the properties and methods of Event as well as the properties and methods defined below

The **ScreenOrientationEvent** has the following properties:

screenWidth This property is of type **Number**

screenHeight; This property is of type **Number**

screenAngle; This property is of type **Number**

softKeysLocation; This property is of type **String**

Appendix E. UAProf Vocabulary Definitions

This section is informative.

E.1 RME Capable

```
<rdf:Description rdf:ID='RMECapable'>

   <rdfs:comment xml:lang='en'>

      Description:   Indicates whether the device provides Rich
                     Media User Agent Services.

      Examples: "Yes", "No"

   </rdfs:comment>

   <rdfs:label xml:lang='en'>RMECapable</rdfs:label>

   <rdf:type rdf:resource='&ns-rdf;Property'/>

   <rdfs:domain rdf:resource='#SoftwarePlatform'/>

   <rdfs:range rdf:resource='&prf-dt;Boolean'/>

   <prf:ResolutionRule rdf:datatype='&prf-dt;
    ResolutionRule'>Locked</prf:ResolutionRule>

</rdf:Description>
```

E.2 RME Version

```
<rdf:Description rdf:ID='RMEVersion'>

   <rdfs:comment xml:lang='en'>

      Description:   Defines version of the Rich
                     Media Enabler (RME) capability within the device.

      Examples: "1.0"

   </rdfs:comment>

   <rdfs:label xml:lang='en'>RMEVersion</rdfs:label>

   <rdf:type rdf:resource='&ns-rdf;Property'/>

   <rdfs:domain rdf:resource='#SoftwarePlatform'/>

   <rdfs:range rdf:resource='&prf-dt;Literal'/>

   <prf:ResolutionRule rdf:datatype='&prf-dt;
        ResolutionRule'>Locked</prf:ResolutionRule>

</rdf:Description>
```

E.3 RME Delivery Contexts

```
<rdf:Description rdf:ID='RMEDeliveryContexts'>

   <rdfs:comment xml:lang='en'>
```

```
      Description:   List of RME Delivery contexts the device supports
                     for transporting RME data streams. Property value
                     is a list of supported contexts, where each item
                     in the list is a context name.

      Examples: "3GP", "http", "RTP"

   </rdfs:comment>

   <rdfs:label xml:lang='en'>RMEDeliveryContexts </rdfs:label>

   <rdf:type rdf:resource='&ns-rdf;Property'/>

   <rdfs:domain rdf:resource='#SoftwarePlatform'/>

   <rdfs:range rdf:resource='&prf-dt;Literal'/>

   <rdfs:range rdf:resource='&ns-rdf;Bag'/>

   <prf:ResolutionRule rdf:datatype='&prf-
dt;ResolutionRule'>Append</prf:ResolutionRule>

</rdf:Description>
```

E.4 Screen Orientations

```
<rdf:Description rdf:ID='ScreenOrientations'>

   <rdfs:comment xml:lang='en'>

      Description:   Describes the list of supported screen
                      orientation capabilities. Note: there
                     is no notion of whether the display is
                     “upside-down” or not. “Portrait” is also
                     0 or 180 degrees. “Landscape” is also
                     90 or 270 degrees.“Angle” implies arbitrary
                     angular rotation.

      Examples: “portrait”, “landscape”, “angle”

   </rdfs:comment>

   <rdfs:label xml:lang='en'>ScreenOrientations</rdfs:label>

   <rdf:type rdf:resource='&ns-rdf;Property'/>

   <rdfs:domain rdf:resource='#SoftwarePlatform'/>

   <rdfs:range rdf:resource='&prf-dt;Literal'/>

   <rdfs:range rdf:resource='&ns-rdf;Bag'/>

   <prf:ResolutionRule rdf:datatype='&prf-dt;
        ResolutionRule'>Locked</prf:ResolutionRule>

</rdf:Description>
```

E.5 ScreenDefaultOrientation

```
<rdf:Description rdf:ID='ScreenDefaultOrientation'>

   <rdfs:comment xml:lang='en'>

      Description:   Describes the default orientation
                     of a device as either “portrait” or “landscape”.
```

```
      Example: “portrait”

   </rdfs:comment>

   <rdfs:label xml:lang='en'>ScreenDefaultOrientation</rdfs:label>

   <rdf:type rdf:resource='&ns-rdf;Property'/>

   <rdfs:domain rdf:resource='#SoftwarePlatform'/>

   <rdfs:range rdf:resource='&prf-dt;Literal'/>

   <prf:ResolutionRule rdf:datatype='&prf-dt;
        ResolutionRule'>Locked</prf:ResolutionRule>

</rdf:Description>
```

E.6 CurrentScreenOrientation

```
<rdf:Description rdf:ID='CurrentScreenOrientation'>

   <rdfs:comment xml:lang='en'>

      Description:   Shows the current screen orientation. Portrait
                     is always long dimension vertical. Landscape is
                     always long dimension horizontal. An angular
                     reference assumes portrait is 0 or 180 degrees
                     and landscape is 90 or 270 degrees.

      Example: “portrait”, “landscape”, “270”, “320”

   </rdfs:comment>

   <rdfs:label xml:lang='en'>CurrentScreenOrientation</rdfs:label>

   <rdf:type rdf:resource='&ns-rdf;Property'/>

   <rdfs:domain rdf:resource='#SoftwarePlatform'/>

   <rdfs:range rdf:resource='&prf-dt;Literal'/>

   <prf:ResolutionRule rdf:datatype='&prf-dt;
        ResolutionRule'>Override</prf:ResolutionRule>

</rdf:Description>
```

E.7 CurrentRMEScreenSize

```
<rdf:Description rdf:ID='CurrentRMEScreenSize'>

   <rdfs:comment xml:lang='en'>

      Description:   Contains the current running screen
                     viewport size for RME in pixels,
                     expressed as width by height.

      Example: "500x300"

   </rdfs:comment>

   <rdfs:label xml:lang='en'>CurrentRMEScreenSize</rdfs:label>
```

```
   <rdf:type rdf:resource='&ns-rdf;Property'/>
   <rdfs:domain rdf:resource='#SoftwarePlatform'/>
   <rdfs:range rdf:resource='&prf-dt;Dimension'/>
   <prf:ResolutionRule rdf:datatype='&prf-dt;
        ResolutionRule'>Override</prf:ResolutionRule>
</rdf:Description>
```