Rich Media Environment Architecture

Approved Version 1.0 – 29 Mar 2011

Open Mobile Alliance

OMA-AD-Rich_Media_Environment-V1_0-20110329-A

Contents

Figures

1. Scope (Informative)

The Rich Media Environment addresses enhanced rich media services: defining the environment from rich-media content and service creation, through deployment, distribution and presentation.

The objective of this document is the definition of the architecture for the rich media environment to meet the requirements defined in the Rich Media Environment Requirements Document [RME-RD].

2. References

2.1 Normative References

[OMA-DICT] "OMA Dictionary", OMA-Dictionary-V1_0-URL: http://www.openmobilealliance.org/ftp/PD/OMA-Dictionary-V1_0-20031014-A.zip

[OSE] "OMA Service Environment" URL: http://www.openmobilealliance.org/

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

[RFC2965] HTTP State Management Mechanism, D. Kristol, L. Montulli, October 2000, URL: http://www.ietf.org/rfc/rfc2965.txt?number=2965

[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*

[SVG] W3C SVG Tiny 1.2 Scalable Vector Graphics (SVG) Tiny 1.2 Specification, Recommendation, http://www.w3.org/TR/SVGMobile12/

2.2 Informative References

[3GPP DIMS] 3GPP Dynamic Interactive Multimedia Scene: *http://www.3gpp.org/ftp/Specs/html-info/26142.htm*

[ARCH-PRINC] "OMA Architecture Principles", OMA-ArchitecturePrinciples-V1_2,URL: http://www.openmobilealliance.org/ftp/Public_documents/ARCH/permanent_documents/OMA-ArchitecturePrinciples-V1_2-20040414-A.zip

[ARCH-REVIEW] "OMA Architecture Review Process", OMA-ORG-ARCHReviewProcess-V1_4, URL: http://www.openmobilealliance.org/ftp/Public_documents/ARCH/Permanent_documents/OMA-ORG-ARCHReviewProcess-V1_4-20060131-A.zip

[OMA-BCAST-AD] "Mobile Broadcast Service", Open Mobile Alliance, OMA-AD-BCAST-V1_0, URL: http://www.openmobilealliance.org/

[OMA-BCAST_Distribution] "Mobile Broadcast Service Distribution", Open Mobile Alliance, OMA-TS-BCAST_Distribution-V1_0_0, URL: http://www.openmobilealliance.org/

[OMA-BCAST_DVB_Adaptation] "Mobile Broadcast service adaptation to DVB", Open Mobile Alliance, OMA-TS BCAST_DVB_Adaptation-V1_0_0, URL: http://www.openmobilealliance.org/

[OMA-BCAST_Services] "Mobile Broadcast Service", Open Mobile Alliance, OMA-TS-BCAST_Services-V1_0, URL: http://www.openmobilealliance.org/

[OMA-BCAST_Service_Guide] "Mobile Broadcast Service Guide", Open Mobile Alliance, OMA-TS-BCAST_Service_Guide-V1_0_0, URL: http://www.openmobilealliance.org/

[OMA-BCAST_SvcCntProtection] "Mobile Broadcast service protection", Open Mobile Alliance, OMA-TS-BCAST_SvcCntProtection-V1_0 URL: http://www.openmobilealliance.org/

[OMA-CP-ARCH] "Content Provisioning", Open Mobile Alliance, OMA-TS-CP-ProvArch-V1_2, URL: http://www.openmobilealliance.org/

[OMA-DCD-AD] "Dynamic Content Delivery", Open Mobile Alliance, OMA-AD-DCD-V1_0, URL: http://www.openmobilealliance.org/

[OMA-DM-AD] "Device Management",Open Mobile Alliance, OMA-AD-DM-V1_0, URL: http://www.openmobilealliance.org/

[OMA-DPE] "Device Profile Evolution"Open Mobile Alliance, OMA-AD-DPE-V1_0 URL: http://www.openmobilealliance.org/

[OMA-FUMO-AD] "Firmware Update Management Object, Open Mobile Alliance, OMA-AD-FUMO-V1_0, URL: http://www.openmobilealliance.org/

[OMA-LOC] "Location", Open Mobile Alliance, OMA-AD-SUPL-V2_0, URL: http://www.openmobilealliance.org/

[OMA-MCC-AD] "Mobile Commerce and Charging", Open Mobile Alliance, OMA-AD-Charging-V1_0, URL: http://www.openmobilealliance.org/

[OMA-PRS-SIMPLE-AD] "Presence Simple" Open Mobile Alliance, OMA-AD-Presence_SIMPLE-V2_0, URL: http://www.openmobilealliance.org/

OMA-PUSH-PPG "Push Proxy Gateway", Open Mobile Alliance, OMA-WAP-TS-PPGService-V2_1, URL:http://www.openmobilealliance.org/

OMA-PUSH-OTA "Push Over The Air Protocol", Open Mobile Alliance, OMA-AD-Push-V2_2, URL: http://www.openmobilealliance.org/

OMA-SCOMO-AD] "Software Component Management Object" Open Mobile Alliance, OMA-AD-SCOMO-V1_0, URL: http://www.openmobilealliance.org/

OMA-SIP-PUSH "Session Initiation Protocol", Open Mobile Alliance, OMA-AD-SIP_Push_AD-V1_0_0, URL: http://www.openmobilealliance.org/

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

[OPENTYPE] ISO/IEC 14496-18, Information technology -- Coding of audio-visual objects -- Part 18: Font compression and streaming, URL: *http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=40151&ICS1=35&ICS2=40&ICS3=*

[RME-WP] OMA Rich-Media Environment White Paper, technology landscape, Open Mobile Alliance, OMA-WP-Rich-Media-Environment : *http://member.openmobilealliance.org/ftp/Public_documents/BAC/MAE/Permanent_documents/OMA-WP-Rich-Media-Environment-20060406-D.zip*

[SVG-FONTS] "SVG Font Format", *URL: http://www.w3.org/TR/SVGMobile12/fonts.html*

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

Interface	See [OMA-DICT].
Delivery context	Infrastructure protocols and packaging that allows the delivery of application data. Note: This is unrelated to W3C definition of Delivery context.
Initial Scene	A scene to which no updates have been applied
Non-timed delivery Context	A delivery context not supplying timing information together with the data
RME data	XML data consisting of either. initial scene data or scene updates
Scene	The description of temporal and spatial layout of objects (included or by references) such as vector graphics, images, audios and their linking to each other
Scene update	Information defining modifications to a scene
Scene Update mechanism	A means to apply scene update to a scene
Timed delivery Context	A delivery context supplying timing information together with the data e.g.: a RTP connection or a 3GP file.

3.3 Abbreviations

3GP File	Third Generation Partnership Project File Format
BCAST	mobile BroadCAST services enabler
CP	Content Provisioning
DVB	Digital Video Broadcasting
DCD	Dynamic Content Delivery enabler
DPE	Device Profile Evolution enabler
DRM	Digital Right Management
FEC	Forward Error Correction
FUMO	Firmware Update Management Object
HTTP	Hyper Text Transfer Protocol
LOC	Location
MMS	Multimedia Message Service
OMA	Open Mobile Alliance
OTA	Over The Air
PAP	Push Access Protocol
PPG	Push Proxy Gateway

RME	Rich-Media Environment Enabler
RTP	Real-Time Transport Protocol
SCOMO	Software Components Management Object
SIP	Session Initiation Protocol
SMS	Short Message Service
TCP	Transmission Control Protocol
UAPROF	User Agent PROFile
UDP	User Datagram Protocol
URL	Uniform Resource Locator

4. Introduction (Informative)

This document defines the architecture for the rich media environment to meet the requirements defined in the Rich Media Environment Requirements Document [RME-RD].

The Rich Media Environment addresses enhanced rich media services: including rich-media content and service creation, deployment, distribution and presentation.

Enhanced rich media services include:

- Services aggregating various kinds of content in a single interface (graphics, text, audio, video) allowing:
  - The rendering and behaviour of the service on the device, or service end-point consuming the service, to meet the developer intent and adapting to the device's characteristics (screen size, input capabilities, etc.)
  - Progressive rendering and fast display of the service content to avoid periods of inactivity waiting for a service response.
  - An intuitive and easy user interaction with the content for navigation.
  - Accurate synchronisation of content, whether streamed or otherwise, and interaction.
  - Receiving data from various networks or bearers simultaneously and presenting the relevant data in a single service interface.
- Service logic based on client-server real-time interaction with:
  - Dynamic content and presentation updates triggered by the service logic and/or by the end-user.
  - The ability to add data at any point in the delivery process, even to an already started progressive download, to decrease end-user latency.
  - Managing data both on client and server side for storage and temporal usage and availability of data in conformance with the logic and delivery of the service.
  - Packaging data in a convenient way to decrease network requests and server resources.

The Rich-Media enabler intends to be usable in the following scenarios:

- Storage of content locally on the device
- downloaded to the device via any file download mechanism
- distributed to the device via point-to-point or point-to-multipoint streams
  - Point-to-multipoint distribution can be in the form of broadcast distribution of data
  - Terminals may join existing broadcast streams. RME solves the problem to tune-in and synchronising to the stream.

The intent of the rich-media environment is to enable services which are network, device, OS and codec independent.

4.1 Planned Phases

The rich media enabler is only planned to be released in one phase and this document describes the complete architecture.

4.2 Security Considerations

The architecture described here contains scripting functionality and a dynamic access mechanism which may require attention from the security perspective; see the section on scripting below (section 5.2.1.1.2)

The architecture does not explicitly call out any security functions provided by other OMA enablers.

Dynamic access to persistent storage uses a security mechanisms based on the http state management.

This enabler may make use of other enablers providing security functionality if deemed necessary.

5. Architectural Model

5.1 Architectural Diagram

The diagram in Figure 1 shows the architecture of the RME Enabler, describing the scope of the RME specification. The Content Provider and other enablers present on the device are also shown.

The architecture is described from a functional/logical view and the diagrams should be interpreted according to the following conventions:

- Boxes describe architectural components that provide a specific functionality. The components are logical and need not coincide with the implementation components.
- Arrows describe interfaces between components and originate from the component initiating the communication.
- Dashed arrows describe potential interfaces between the RME enabler and other applications or OMA enablers. These interfaces are implementation dependent.

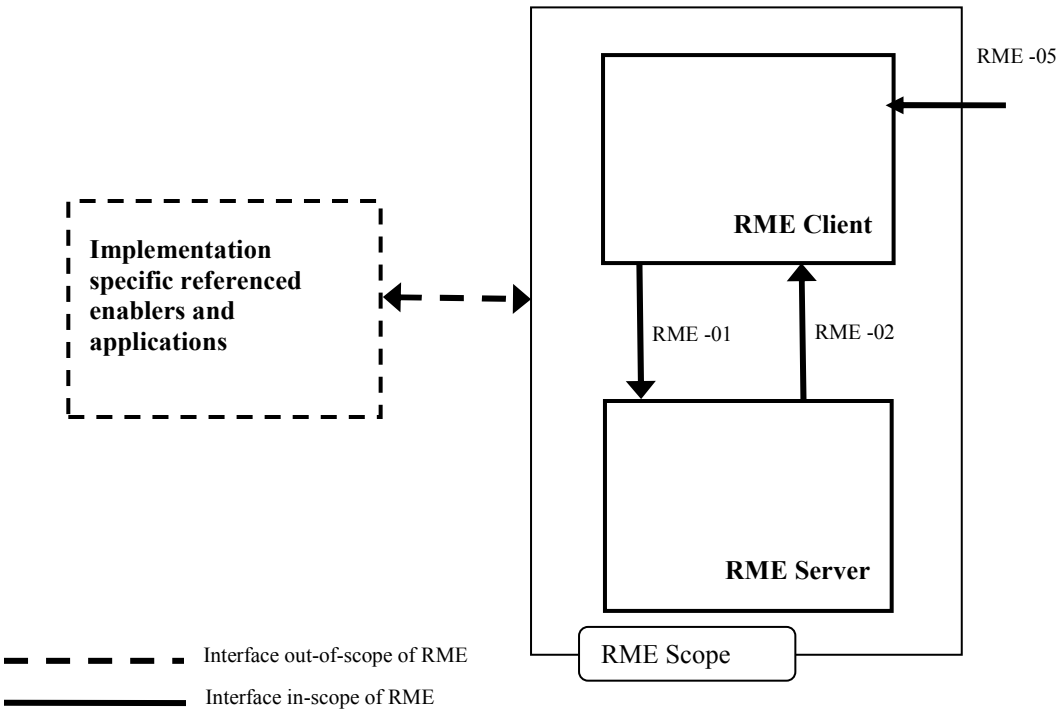

Figure 1: Architecture of the RME Client and Server including the scope of RME

In Figure 2, the functional composition of the RME Client is shown

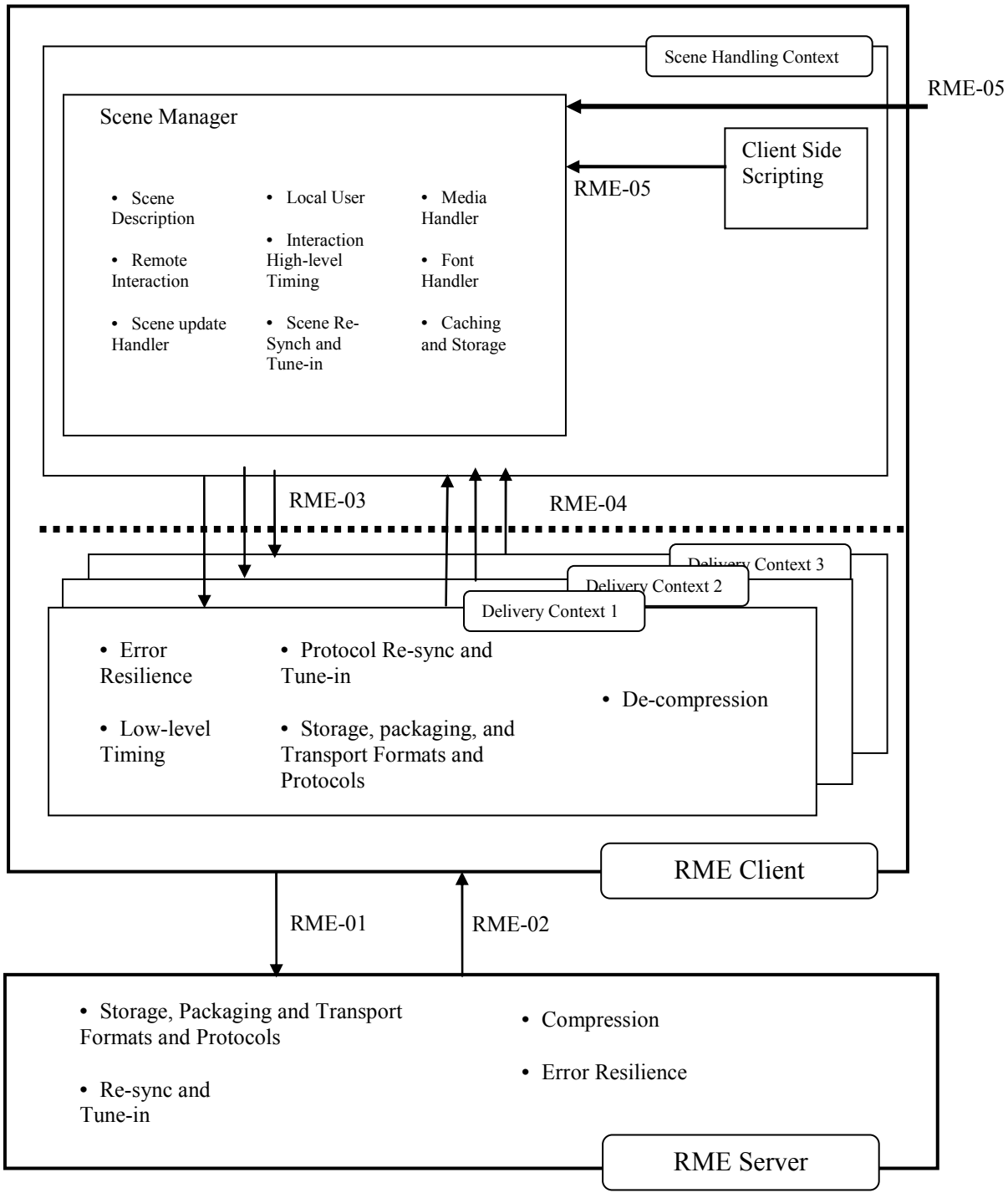

Figure 2 Functional compositions of the RME Client and the RME Server

5.2 Functional Components and Interfaces

The RME Enabler consists of the RME Client and the RME Server. As the RME Enabler is agnostic to transport protocols the communication between the server and client may utilize different protocols such as UDP, TCP, HTTP etc.

The client and server components are described in detail in the following sections.

5.2.1 RME Client

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 handling interaction with the client itself (local interaction) and with the RME server (remote interaction).

The RME Client is divided into two parts: Scene Handling Context and Delivery Context. A number of different delivery contexts can be used to deliver the RME data to and from the server.

Requests and receipt of RME data between the Scene Handling Context and different Delivery Contexts is performed via interfaces RME-03 and RME-04. RME-05 is the interface between the scripting module and the main component of the RME Client, the Scene Manager.

5.2.1.1 RME Client Components – Scene Handling Context

The Scene Handling Context is defined as the component which interprets the RME data and associated media data and generates the displayed content. As further described below it provides the functionality for synchronization of the RME scene.

The Scene Handling Context is composed of two components:

- Scene Manager
- Client Side Scripting

5.2.1.1.1 Scene Manager

The Scene Manager is the main component of the RME Client. It is composed of the following functions:

- **Scene Description** - handles the description and rendering of the graphics constituting the RME Scene. It is responsible for the spatial layout, compositing and rendering of an RME Scene and acts as the Scene container by referencing or including other media data.
- **Local User Interaction** - enables the user to locally interact with the rich media content. It comprises of an event manager that enables the management of event creation, dispatching of events, registering and invoking the event listeners.
- **Remote Interaction** - handles the client-server communication. It is used for setting up data sessions, requesting data, transferring user data and events from the client to the server via RME-03.
- **Media Handler** – responsible for handling the relevant decoders supported by the terminal for raster image formats, as well as for the different audio and video streaming formats transmitted along with RME scene and data..
- **High-level Timing** – handles the timing and synchronization of media elements present in the RME scene. The synchronization can be between any objects in the scene and it can also be between the scene and the application of updates to the scene.
- **Font Handler** – handles the fonts supported by the RME Enabler. The RME Enabler SHALL support downloadable fonts as a part of rich media content. The RME enabler is strongly recommended to support both the OpenType font

format [OPENTYPE] and the SVG font format [SVG-FONTS]. RME SHOULD support a minimum set of metrically compatible scalable resident fonts.

- **Scene Update Management** – applies the received updates to the current scene. Scene updates can be delivered to a client from a server without the client requesting it, or they can be requested from the client in a client pull scenario, either initiated by a user action or through timing. The Scene Update Mechanism receives the Scene Update via RME-04.

  See section 5.3 Flows, for more detail on the different update scenarios.

- **Scene Re-Synch and Tune-in** – handles the tune-in and re-synchronization on the scene level.

- **Caching and storage** – handles the local storage of data e.g. user preferences and service data. The security consideration for cashing and storage should at a minimum conform to [RFC2965].

5.2.1.1.2 Client Side Scripting

The RME Client contains a module for handling Client Side Scripting. It enables procedural manipulations of the Scene and provides a powerful application development environment to the content creator. Communication between the Scene Manager and the Client Side Scripting is performed via RME-05.

Running of scripts on the client may constitute a security risk. The specified security mechanism should as a minimum impose the same level of security as is required for XHTML browsing environment and be compatible with the cookies access restrictions described in [RFC2965].

The technical specification will define a security mechanism that constitutes a minimum level of security, to which the RME enabler must conform.

5.2.1.2 RME Client Components – Delivery context level

The included list describes the functions of the RME Enabler on the Delivery context level. These functions may be achieved by reusing the work of other OMA enablers or external services. Its use will be defined within the OMA RME technical specification. Hence, this section only contains a brief overview of the functions that will be necessary.

Note: all these modules will not be necessary in all delivery contexts, e.g. Re-synchronization and Tune-in will not be necessary in a reliable delivery context.

- **Low-level Timing** – In cases where the transport protocol or the container format includes timing information, the Low-level Timing module extracts the timing information and passes this to the application level for use in the High-level Timing model. For example, in the case of RTP, the RTP timestamp, sequence numbers etc, associated with a data packet are sent to the scene handling context for synchronization. In the case the media is packaged in a 3GP file (resident on the device, downloaded or progressively downloaded) the timing information in the 3GP file is extracted and delivered to the application layer together with the associated data.

- **Protocol Re-synchronization and Tune-in** – handles re-synchronization of a stream that has gone out of sync due to packet loss and/or tunes in to a broadcast or unicast rich media stream.

- **Error Resilience** – handles error recovery when using unreliable delivery protocols using e.g. FEC and re-transmission techniques.

- **De-compression** – handles de-compression of the compressed RME data if compression was used or if decompression is needed.

- **Storage, Packaging and Transport Protocols/Formats** – handles protocols and formats in order to store and deliver the RME data to the client, e.g. RTP payload formats etc.

5.2.2 RME Server

An RME Server supplies the RME Client with rich media data. The data specific to RME are Initial Scenes and Scene Updates. Other data such as script files, images and audio and video may be provided by the RME server to compose the rich media service. Initial Scene, Scene Updates, scripts and media can be referenced by the scene description (via dedicated attributes/elements, e.g: video element) from different servers. They can be synched to the RME-scene timeline and each other using high level and low level timing components

An RME Server that creates RME data on the fly SHALL conform to the client-server interfaces and data formats transported on the client server interfaces RME-01 and RME-02.

The RME server functions are:

- **Protocol Re-synchronization and Tune-in** – functionality that provides tune-in points in the data flow.
- **Error Resilience** – functionality that provides error resilience data to the client.
- **Compression** – Functionality that provide the compression of the RME data.
- **Storage, Packaging and Transport Protocols/Formats** – handles the protocols and transport formats.

5.2.3 RME Interfaces

This section describes the interfaces between the components in the RME architecture. The interfaces are shown in Figure 2. The descriptions identify which information is transferred and when necessary, the format used to transport the information is also identified.

It is possible to implement RME on devices without a back channel; however, interactivity will be limited to local interactions.

Devices without backchannels MUST handle backchannel communication gracefully.

5.2.3.1 RME-01 RME Server Interface

This interface is used for download scenarios and point to point streaming (multicast) scenarios.

This interface is used to provide data to the RME server (consisting of data requests and settings of preferences) and return data from the RME server (consisting of initial scene data, scene updates and data used to populate the scene objects).

The RME enabler is bearer agnostic and the information to the RME server can be exchanged using various packaging and storage format depending on the delivery context and bearers.

5.2.3.2 RME-02 –Broadcast and Push Interface

Communication on the RME-02 interface is recieved by the RME client.

The interface is used for broadcast use cases and for push use cases

The data provided, through this interface, are initial scene data, scene updates and data used to populate the scene objects.

Note: the expectation is that any negotiation for the use of the service is done by other means.

5.2.3.3 RME-03 Delivery Context Interface

The RME-03 interface is exposed by the delivery context.

This interface is used by the Scene Manager to deliver information such as:

- Sending user data to the server
- Sending preferences to the server
- Requesting data from server such as scene updates, scene object data or RME data.
- Requesting a new channel
- Requesting a new data channel
- requesting synchronization of an existing data channel

5.2.3.4 RME-04 Scene Handling Context Interface

RME-04 is an interface exposed by the the Scene Handling Context.

The interface is used for transferring:

- Initial Scene data which contains the description of the initial scene.
- Scene Updates from the delivery context to the Scene Update Handling.
- Scripts to the Client Side Scripting.
- Data used by Scene objects containing other media such as video, audio and still images.

5.2.3.5 RME-05 Scene Manager Interface

RME-05 is an interface exposed by the Scene manager, allowing script delivered to the client to manipulate the Scene. The interface is defined as an extension of the SVG uDOM [SVG] and is an interface to the Scene DOM-tree. This interface can also be used by external components (e.g. browsers) to interact with the RME client.

5.2.4 Timing model and processing model

The timing model handles the timing and synchronization of the RME scene. The synchronization can be between any objects in the scene and it can also be between the scene objects and the application of updates to the scene. All synchronization is performed relative to Document Time, which is the master time. The Document Time is set to 0 each time an Initial Scene is loaded and advances according to the SVGT1.2 specification [SVG]

Synchronizable rich-media elements, available in the media type, SHALL conform to the W3C SVGT1.2 specification [SVG].

The delivery of timing information for video and audio used to synchronize these objects is in accordance with the particular format used for each object. Details of these formats are not in the scope of the rich media enabler specification.

Updates do not contain intrinsic timing information and thus need the timing information to be supplied, to allow synchronization of single or multiple updates with the scene. It MUST be possible to handle the following timing scenarios:

- Updates are streamed to the device and applied synchronized to the scene time.
- Updates are stored on the device or downloaded to the device and applied synchronized to the scene time
- Updates are stored on the device or downloaded on the device and applied as soon as possible.

The technical specification SHALL specify a processing model detailed enough that the implementations applying the processing model will yield only one correct version of the scene.

5.2.5 Interfaces to and from other Enablers

This section describes the relationship between the RME enabler and other OMA enablers with which the RME enabler can interact or that would provide support to the RME enabler.

5.2.5.1 Browser

The browser has href link support, which will allow for the processing of the RME mime type if supported by the system.

The RME engine does support the processing of href. If a URL mime type is received, which is processed by the browser user agent, the browser will be invoked.

5.2.5.2 DRM

RME can make use of DRM to encrypt and decrypt content.

RME can make use of DRM for managing access to content.

5.2.5.3 BCAST

The BCAST enabler can be used in multiple ways along with the RME enabler.

Interactions between the RME and BCAST enabler are as follows:

- The RME client may be used as the presentation layer of the BCAST Service Guide.
- Service Guide Function: BCAST Service Guide can be used to provide users with information about RME Services or Content. [OMA-BCAST_Service_Guide], [OMA-BCAST_Services]
- File / Stream Distribution Function: BCAST File and Stream Distribution Functions can allow respectively efficient file and streaming delivery for the broadcast of RME contents to RME Clients. [OMA-BCAST_Distribution]
- Service and Content Protection Function of BCAST can be used either to control access to RME content and RME stream or only to associated AV streams. [OMA-BCAST_SvcCntProtection]

An RME/BCAST adaptation specification may be needed during the Technical Specification Phase to allow and clarify the combined usage of RME and BCAST in particular over DVB bearer [OMA-BCAST_DVB_Adaptation]. Some interfaces are already defined in the BCAST specification [OMA-BCAST-AD], some other may be needed.

5.2.5.4 Push

If RME content notification is pushed from a server to a client, the Push Enabler may be used for delivery of content or administration actions to the RME Client either by interacting with:

- The Push Proxy Gateway (PPG) entity via the Push Access Protocol (PAP) interface defined in [**Error! Reference source not found.**] for adaptation of Push operations to underlying bearers
- The Push Client entity via the Push Over-the-Air protocol (Push-OTA) interface defined in [**Error! Reference source not found.**]
- The SIP Push Client entity via the Push over SIP protocol interface defined in [**Error! Reference source not found.**]

The RME Client uses the Push Enabler for reception of content notification from the Push Client.

5.2.5.5 UAProf

UAProf vocabulary will be required to identify and specify RME capabilities.

5.3 Flows

5.3.1 Sub Use Cases

The requirements document [RME-RD] for RME in OMA contains the following use cases:

- *I&E, Karaoke*
- *P2P, live chat, rich-media blog service*
- *I&E, Interactive mobile TV services (mosaic, interactive voting service, personalized menu)*
- *Active wall paper services*
- *Rich mobile application*

The use cases have some functionality in common. For instance all use cases load a rich media scene. By creating a sub use case for loading a scene and let all use cases point to this, it is sufficient to describe the flow for loading a scene once. By analyzing the use cases a set of sub use cases are constructed. By combining these sub use cases in various ways all use cases in the RD can be realized. Below is a list of the identified sub use cases (SUB). Detailed description and flows of the sub use cases follows in the next section.

SUB 1 **Loading of an Initial Scene.** This use case covers loading of an Initial Scene either from local storage or via a network.

SUB 2 **Single source delivery.** This use case covers the loading of an Initial Scene and accompanying Updates from one source.

SUB 3 **Multi source delivery.** This use case handles the setup of an update source from which Scene Updates are retrieved.

SUB 4 **Local interaction.** This use case covers interaction with the Scene handled locally on the device e.g. user interaction or timed interaction using script. Interaction with other enablers on the device e.g. the SMS application is also included in this use case.

SUB 5 **Remote interaction.** User interaction with the server causing changes to Scene and/or changes the state on the server.

SUB 6 **Server push.** The server notifies the client that there is data to retrieve.

5.3.1.1 Loading of an Initial Scene

The sub use case "Loading an Initial Scene" is initiated by some event, typically a user event, and ends when the RME scene is displayed on the device. The data describing the scene can be retrieved from any accessible source e.g. a file on the local file system, a file downloaded from a server, a unicast or multicast streaming. The source may only contain an Initial Scene and not Scene Updates. The scene could however contain video objects and other referenced objects that contain their own communication channels and thus are updated although the scene is the same. In the case of streams where tune-in is necessary this is handled by the *Delivery Context*. Figure 3 below shows the interaction between the architectural components.

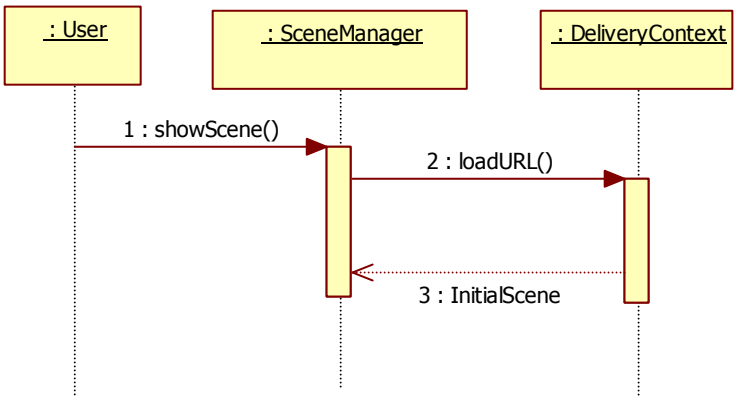

Figure 3: Flow: loading of an initial scene

The *Scene Manager* is invoked with a request to show a Scene based on data from a specified URL. The URL can point to any source described above. The delivery context both at the client and server sides, manages the transfer of the request and the response to the URLs. The delivery mechanism retrieves the data and delivers it to the *Scene Manager* which performs the processing and rendering of the Initial Scene.

5.3.1.2 Single source delivery

The single source delivery flow starts with an action causing the *Scene Manager* to tune-in to a data source containing an Initial Scene and Scene Updates and does not end until the connection is stopped. The number of Scene Updates and Initial Scenes in the data is arbitrary. The flow is valid both, when the data is obtained from a container file or a regular file located either on the device or on a RME server and when the data is obtained from a unicast or broadcast stream. The figure below shows the interaction between the different functional components. The difference from the Loading of an Initial Scene use case is that the data channel is kept open to receive updates after the initial scene is loaded whereas it was closed in the previous use case.

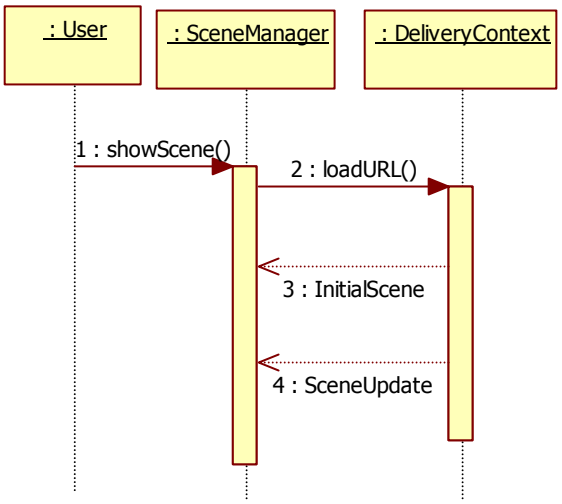

Figure 4: Flow: single source delivery

The flow starts when the *Scene Manager* is invoked with a request to show RME data based on data from a specified URL. The URL can point to any source as described above. The *Scene Manager* calls *Delivery Context* (both client and server side) with a request to tune-in to the given URL. The *Delivery Context* processes the stream to find a tune-in point. The scene contained in the tune-in point is forwarded to the *Scene Manager*, which displays the Scene. *Delivery Context* continues to analyze the incoming data and forwards incoming Scene Updates and Scenes and associated media data to *Scene Manager*. In case of data losses re-synchronization is performed by the *Delivery Context* which forwards data contained in a successive tune-in point to the *Scene Manager*.

5.3.1.3 Multi source delivery

This flow assumes that a Scene is already loaded and displayed by the *Scene Manager*. The Scene can be loaded due to either the sub use case "Single Source Delivery" or "Loading of an Initial Scene". The flow start when some action, possibly by an user, causes the *Scene Manager* to retrieve RME data from a specified URL. The URL can point to any file local or remote or to a unicast or broadcast stream. The flow ends when all Scene Updates are consumed and/or the connection to the source is closed possibly by the end user. The figure below shows the interaction between the different functional components.

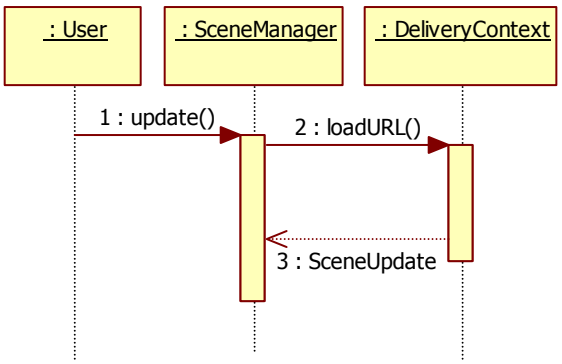

Figure 5: Flow: multi source delivery

The *Scene Manager* commands *Delivery Context* (both client and server side) to retrieve data at a certain URL. The URL can be any source described above. *Delivery Context* extracts the data and forwards it to *Scene Manager*, which applies the changes to the RME scene.

5.3.1.4 Local interaction

The flow for local interaction assumes that the *Scene Manager* contains and displays a Scene. The flow starts when some action causes the *Scene Manager* to interact with either itself or another enabler on the device. The figure below shows four different cases for local interaction:

- Actions handled by the *Scene Manager* itself e.g.: events (1)
- Actions that causes a script to execute and via the uDOM affect the *Scene Manager* (2-3)
- Actions that cause a script to execute which uses other enablers on the device (4-5)
- Actions that invoke other enablers on the device without script (6)

The list of possible actions is not exhaustive.

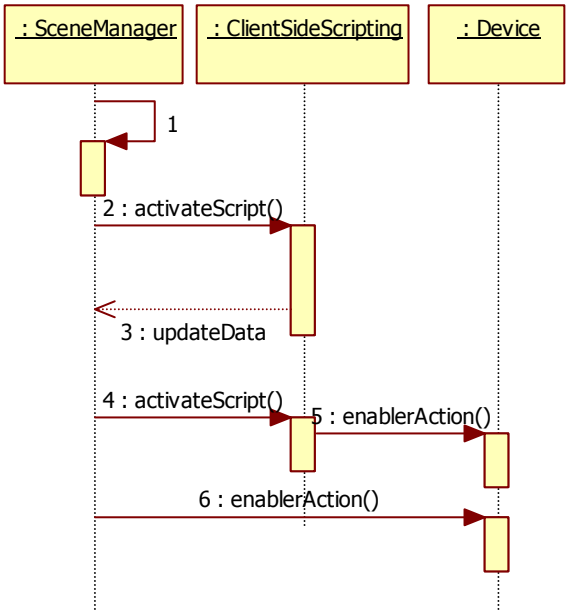

Figure 6: Flow: local interaction

5.3.1.5 Remote interaction

This flow assumes that a Scene is loaded and displayed by the *Scene Manager*.

The figure below shows two examples of a Remote interaction between the functional components.

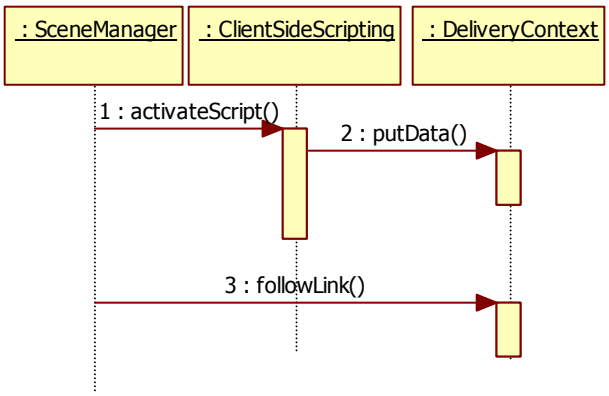

Figure 7: Flow: remote interaction

Some action causes the *Scene Manager* to activate a script which sends data to the server.

5.3.1.6 Server Push

The flow for server push requires that the RME enabler is registered as a listener to a port. The flow starts when the server pushes data to the client and ends when the client has taken the action required in the push message. The figure shows a push message requiring loading of a new scene.

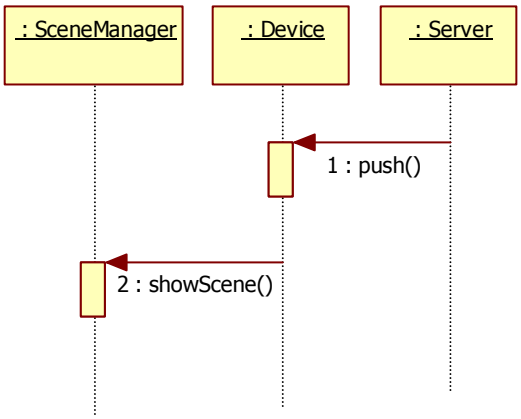

Figure 8: Flow: server push

5.3.2 Flows of the High-level Use Cases

This section shows how the high level use cases in the requirements document realized by the sub use cases. Further analysis can then be focused on the sub use cases.

5.3.2.1 Use Case: I&E, Karaoke

SUB 1: Loading of an Initial Scene (Selection of the service)

SUB 4: Local interaction (Browsing of the song catalogue)

SUB 2, SUB 3, and/or: Single source delivery, Multi source delivery (Provision of content from service provider)

SUB 5: Remote interaction (Selects language)

SUB 2 or SUB 3: Single source delivery or Multi source delivery (Display and synchronization of content)

SUB 4: Local interaction (Changing the volume)

SUB 5: Remote interaction (Media control)

SUB 4: Local interaction (Send SMS and receive MMS – receive MMS is outside of the RME enabler)

SUB 4: Local interaction (Saving data on the device)

5.3.2.2 Use Case: P2P, live chat, rich-media blog service

SUB 1: Loading of an Initial Scene (Receives an MMS with a link to the service)

SUB 4 and SUB 5: Local interaction (Enters personal information) and Remote Interaction (Uploads and stores the personal information on the server)

SUB 5: Remote interaction (Browsing and selection of data)

SUB 4: Local interaction (Interfacing with the SMS application)

SUB 5: Remote interaction (Send SMS)

SUB 4and SUB 5: Local interaction (Enters chat messages) and Remote Interaction (Sends chat messages)

SUB 2, SUB 3, or SUB 5: Single source delivery, Multi source delivery, Remote Interactions (Retrieve alerts of incoming messages)

SUB 5: Remote interaction (Change chat channel)

SUB 5: Remote interaction (Make phone call through service provider – service specific)

SUB 5: Remote interaction (Add/remove contact from server database)

SUB 2, SUB 3, or SUB 5: Single source delivery, Multi source delivery, and/or Remote Interactions (Retrieve list of active chatters)

SUB 4and SUB 5: Local interaction and Remote Interaction (Validation of user data)

Alternative Flow:

SUB 1, SUB 4 and SUB 5: Loading of a scene, Local interaction and Remote interaction (User requests to chat)

SUB 4: Local interaction (User enters message)

SUB 5: Remote interaction (Message is sent over HTTP to the service provider)

SUB 3 or SUB 5: Multi source delivery or Remote interaction (Service provider sends the message out to the receiver – HTTP assumed)

5.3.2.3 Use Case: I&E, Interactive mobile TV services

Mosaic main page:

SUB 1: Loading of an Initial Scene (The user accesses of the menu)

SUB 3, or SUB 5: Multi source delivery, and/or Remote interactions (The menu is updated with new data)

SUB 4: Local interaction (Updates are stored locally)

SUB 5: Remote interaction (Channel selection)

SUB 4, SUB 5: Local interaction or Remote interaction (Storage of user data)

Interactive voting service:

SUB 2, SUB 3, SUB 4, and/or SUB 5: Single source delivery, Multi source delivery, and/or Remote interactions and/or Local interaction (Synchronization of video and text)

SUB 2, SUB 3, or SUB 5: Single source delivery, Multi source delivery, or Remote interactions and/or Remote interaction (Retrieval of alert)

SUB 1: Loading of scene (The user access a rich media service)

SUB 4 or SUB 5: Local interaction or Remote interaction (Reduction of the video area)

SUB 5: Remote interaction (Channel selection)

SUB 2, SUB 3 and/or SUB 5: Single source delivery, Multi source delivery, and/or Remote interactions and/or Local interaction (Appearance of two buttons)

SUB 4 and SUB 5: Local interaction and Remote interaction (User selects a button, data is sent to service provider using HTTP)

SUB 4: Local interaction (Buttons disappear)

SUB 4 or SUB 5: Local interaction or Remote interaction (Full-screen video)

SUB 2, SUB 3, or SUB 5: Single source delivery, Multi source delivery or Remote interactions and/or Remote interaction (Retrieval of vote result)

SUB 2, SUB 3, SUB 5 and/or SUB 4: Single source delivery, Multi source delivery and/or Remote interactions and/or Local interaction (Appearance of a SMS button)

SUB 4: Local interaction (Writes and sends SMS)

SUB 2: Single source delivery (Delivery of text and images in a single connection)

Personalized menu:

SUB 4 and SUB 5: Local interaction or Remote interaction (Browsing of menu, selection and personalization)

SUB 4: Local interaction (Storage of data)

SUB 2, SUB 3 or SUB 5: Single source delivery, Multi source delivery or Remote interactions and/or Remote interaction (Retrieval of news data, including personalization of the data)

SUB 5: Remote interaction (User data saved on server)

SUB 4: Local interaction (Storage of data)

5.3.2.4 Use Case: Active wall paper services

SUB 1: Loading of an Initial Scene (Retrieval of the application from network or locally)

SUB 4, SUB 5: Local Interaction, Remote Interactions and Local interaction (Use of local and remote resources to set up service)

SUB 2, SUB 3, or SUB 5: Single source delivery, Multi source delivery and/or Remote Interactions (Retrieve content)

SUB 5: Remote interaction (Notification of new data from the server)

SUB 5and SUB 4: Remote interaction and Local interaction (Customization of the service)

5.3.2.5 Use Case: Rich mobile application

SUB 1: Loading of an Initial Scene (Download and start the application)

SUB 3, SUB 4 or SUB 5: Multi source delivery and/or Remote Interactions (Retrieve content)

SUB 4: Local interaction (Browse the content locally)

SUB 5or SUB 6: Remote interaction or Server Push (Notification of new data from the server)

SUB 5and SUB 4: Remote interaction and Local interaction (Customization of the service)

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-AD-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