Unified Virtual Experience

Approved Version 1.0 – 01 Dec 2015

Open Mobile Alliance

OMA-ER-UVE-V1_0-20151201-A

Contents

Figures

Tables

1. Scope

This document provides the architectural and technical aspects for UVE enabler. It provides the basic architecture with different components and interfaces involved. It also defines the functionalities supported on the specified components and interfaces required to enable several requirements as defined in [UVE-RD].

2. References

2.1 Normative References

[AV-RTP] "RTP Profile for Audio and Video Conferences with Minimal Control, RFC 3551", IETF, URL:http://datatracker.ietf.org/doc/rfc3551/

[MoverRTP] "RTP Payload Format for MPEG1/MPEG2 Video", IETF, URL:http://datatracker.ietf.org/doc/rfc2250/

[MpegTS] "Generic coding of moving pictures and associated audio information: Systems",1SO/IEC 13818-1:2000/Amd.3:2004, URL:www.iso.org

[Offer-Answer] "An Offer/Answer Model with Session Description Protocol (SDP) RFC 3264", IETF, URL:http://datatracker.ietf.org/doc/rfc3264/

[OSE] "OMA Service Environment", Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[PUSH-MSG] "Push Message", Open Mobile Alliance™, OMA-TS-Push_Message-V2_3, URL:http://www.openmobilealliance.org/

[PUSH-OTA] "Push Over The Air", Open Mobile Alliance™, OMA-TS-PushOTA-V2_3, URL:http://www.openmobilealliance.org/

[PUSH-PAP] "Push Access Protocol", Open Mobile Alliance™, OMA-TS-PAP-V2_3, URL:http://www.openmobilealliance.org/

[PUSH-SL] "ServiceLoading", WAP Forum™, WAP-168-ServiceLoad-20010731-a, 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

[RFC2616] "Hypertext Transfer Protocol – HTTP/1.1", R.Fielding et al, June 1999, URL:http://www.ietf.org/rfc/rfc2616.txt

[RFC4234] "Augmented BNF for Syntax Specifications: ABNF". D. Crocker, Ed., P. Overell. October 2005, URL:http://www.ietf.org/rfc/rfc4234.txt

[RTP] "RTP: A Transport Protocol for Real-Time Applications", IETF, URL:http://datatracker.ietf.org/doc/rfc3550/

[RTSP] Real Time Streaming Protocol (RTSP), RFC2326

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

[TAS] "Telecom Application Store", Open Mobile Alliance™, URL:http://www.openmobilealliance.org/

[UVE-RD] Unified Virtual Experience requirement document, OMA-RD-UVE_RD-V1_0, OMA, URL:http://www.openmobilealliance.org/

2.2 Informative References

[OMADICT] "Dictionary for OMA Specifications", Version x.y, Open Mobile Alliance™, OMA-ORG-Dictionary-Vx_y, URL:http://www.openmobilealliance.org/

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

Invitee	Invitee is a UVE user who is invited by other UVE user (inviter) via UAInvite Request to consume the UA session.
Joinee	Joinee is a UVE user who is asked to join an existing UA session by other UVE user via Join Request.
Local Resources	Local Resources are the various physical or logical entities, available to the devices, which need to be consumed by the UAs to perform their functions. Local Resources includes sensors (GPS, accelerometer etc.), camera, calendar, address book etc.
No-Loss Reconnection	No-Loss Reconnection is a reconnection option used to resume an unexpected UA session disconnect. If opted it will allow users to start using UA from the same point where it was disconnected.
OS-Independent UAs	OS-Independent UAs refers to those UAs which users can use irrespective of the device OS they are using.
Output Stream	Output Stream is a video stream generated by VM by capturing UA display/sound periodically and putting them in a sequence to get a single audio-video stream. Output Stream MAY be transferred from VM to UC using TS Stream [MpegTS] over RTP [MoverRTP].
Reconnection Method	Reconnection Method is how UVE entities will behave in case of an un-expected session disconnect. There are two type of Reconnection Method defined in this enabler. 1. Normal re-connection: This type of reconnection allow user to reload the UA as per application logic. 2. No-loss re-connection: As per this type of re-connection no user data must be lost. User should start consuming the application from the point where he/she was disconnected.
User Interaction	It consists of user interaction on the UA. It can be keyboard events, mouse click/drag, accelerometer etc.
UVE Application	UVE Application is an application hosted remotely in the network.
Virtual Input Module	Virtual Input Module (VIM) is a logical module, in the VM, corresponding to the each input device connected to the VM for a particular UA. It works as bridge between UA and real input devices connected to the terminal and does adaption for the input devices when needed.
VM Data	VM data is a VM snapshot. Snapshot is a state of a VM at a particular point of time.
Watchee	Watchee is a UVE user with an existing UA session identified by another UVE user (Watcher) to be monitored.
Watcher	Watcher is a UVE user who is authorized to monitor and/or suspend an existing UVE application session.

3.3 Abbreviations

CP	Content Provider
OMA	Open Mobile Alliance
SP	Service Provider
UA	UVE Application

4. Introduction

Traditionally, Applications used by users are installed and running in local devices. Unified Virtual Experience (UVE) can enable users to use applications hosted remotely. Unified Virtual Experience (UVE) hosts various applications (enabling different contents and services) in a unified platform (cloud computing platform), and provides them to UVE users using virtualization techniques (cloud computing). UVE users can use remote applications irrespective of the device platforms they are using with consistent user experiences as compared to using applications hosted locally on the devices.

UVE enabler consists of different actors namely UVE Users, UVE SP, Content Provider (developers) and Application Server. Content provider provides UAs to UVE SP.UVE SP hosts those UAs in their network forming a cloud based services. UVE Users access these services from their terminals using cloud computing technologies.

Figure 1 shows the actors mainly involved in the UVE ecosystem.

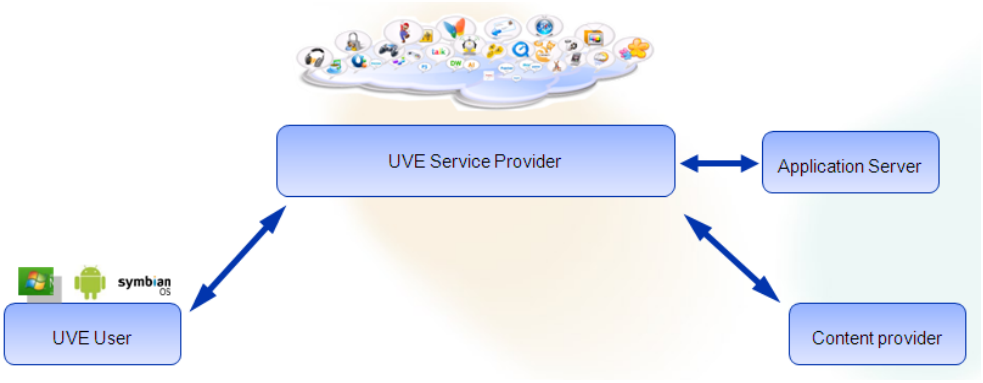

Figure 1: Actors involved in UVE ecosystem

All the depicted actors get benefits from this new UVE paradigm.

UVE User enriches his/her application usage experience by using UAs irrespective of the device OS or device they are using, and using UAs online instead of downloading them on to the device.

Content Provider cut the cost of producing application without creating a different version (for each mobile OS) of a single application.

Service provider offers a larger selection of applications and services to their end-users via UVE Enabler.

The UVE Enabler defines an overall framework that assures remotely located applications used by different devices with UVE capability via cloud computing technology. The UVE User's application usage experience is consistent with locally hosted application.

4.1 Version 1.0

The version 1.0 of the UVE Enabler defines an overall framework that enables Unified Virtual Experience services.

The core functionalities exposed by the UVE Enabler include:

- OS-Independent UAs: This functionality enables users to use UA irrespective of the OS they are using. This will increase the UA availability for users and vice versa.
- On-line UAs: This functionality allows users to use applications online instead of downloading them on to their devices. This will allow users to overcome terminal barriers (e.g memory) and accessing UA from any terminal at any time.
- UA Monitoring: This functionality enables user to monitor an existing UVE application session and if allowed to suspend the monitored session.
- No-loss reconnection: This functionality enables users to reconnect to an abnormally suspended UVE application session without UA data being lost.

5. Requirements (Normative)

UVE requirements are specified in a separate document which can be found at [reference to latest OMA-UVR-RD].

6. Architectural Model

6.1 Architectural Diagram

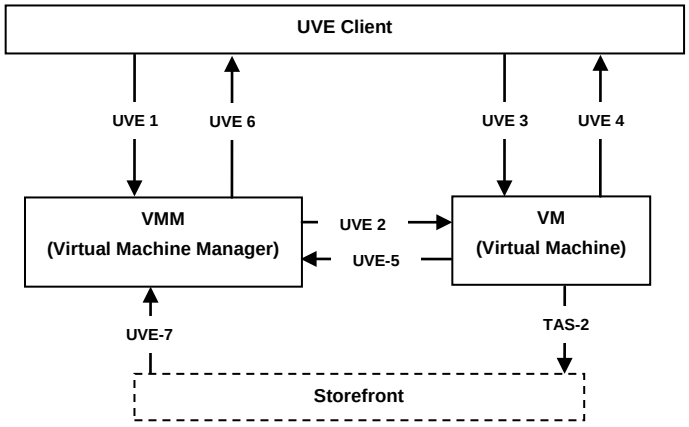

Figure 2: Architectural Diagram

6.2 Functional Components and Interfaces/reference points definition

6.2.1 Functional Components

6.2.1.1 UVE Client (UC)

UC is a device side component residing in the terminals enabling UVE enabled services and utilizing virtualization technology to enable underlying operating system agnostic applications. The UC is mainly responsible for:

- Output rendering: On receiving Output Stream form VM, UC renders Output Stream to the user.
- Interaction provisioning: UC is also responsible to transfer interactions commands to the VM, where they get executed on the UA.
- Local Resource Provisioning: UC is responsible for providing local resource to VM as per the UA requirements. UC may use local device APIs to get the Local Resource data.

6.2.1.2 UVE Server (US)

US is network entity providing unified virtual experience. US comprises of two different components; Virtual Machine Manager (VMM) and Virtual Machines (VMs).

6.2.1.2.1 Virtual Machine Manager

Virtual Machine Manager is a US component responsible for:

- Managing VM(s): VMM is responsible for VM management including deploying UA on VM, delete UA on VM and update UA on VM.

- Selecting VM: On receiving a UA session request from UC, VMM SHALL selects an appropriate/available VM. VM selection will be based on UA identification and terminal capabilities information provided in the UA request from UC.
- Watcher Management: VMM is responsible for Watcher management including enabling Watcher assignment (with different privilege) and Watcher privilege update. In addition it also ensures that at one point of time only one Watcher, among several Watchers of a single application session, shall be able to suspend the monitored application session.
- VM Data store/restore: In case of an exception notification received from VM, VMM is responsible for storing the VM data as per the Reconnection Method specified.
- User Management: VMM is responsible for management of user authentication, authorization, user equipment information and access network information.
- UA Management: VMM is responsible for UA management. Related UA management functionalities can be realized by re-using existing services.
- UA Procurement: VMM is responsible for selecting a suitable VM according to the application meta-data received by storefront on UVE-7 and directing the selected VM to the appropriate storefront to download the application.

6.2.1.2.2 Virtual Machine

Virtual Machine is a US component emulating different operating systems and responsible for:

- Application Hosting: The basic responsibility of VM is to host the UA. The UA can be hosted on VM dynamically at runtime or they can be pre-configured.
- Output generation: VM is the entity which will actually interact with UC in the UA session. One of the main responsibilities of VM is to capture the UA display periodically and creating a single video stream (Output Stream) to be transferred to UC. Where, that will be rendered to the user by UC.
- Local Resource Provisioning: VM is responsible to provide Local Resource data (received from UC) to the UA
- Interaction execution: VM is responsible to execute interaction commands on the UA.
- UA Procurement: VM is also responsible for procuring application (as UAs) from existing storefronts using mechanism provided by storefront itself.
- VM Data deployment and update: VM is responsible for retrieving VM Data for restoring unexpectedly disconnected UA session. Upon user finish consuming the VM or un-expected session disconnection, VM is responsible for updating the VM Data stored in the VMM.

6.2.2 Interface Definition

6.2.2.1 UVE-1

This interface is exposed by VMM and used by UC. The functionalities supported on this interface are:

- UA request: This interface is used by UC to request the desired UA from US. A set of information will be provided in the request including UA identification, terminal capabilities, user identification, Reconnection Method requested etc. VMM will response back with an appropriate VM to which the UC should connect in order to use the requested UA. VMM will select the appropriate VM as defined in section 7.2.2.2.
- Monitoring request: This interface is used by UC to send a monitoring request to the VMM. A unique identification of a Watchee (to-be monitored user) will be provided in this request.
- Session suspend request: This interface is used by UC (as Watcher) to suspend a monitored UA session. Privilege check for Watcher is pre-requisite for suspending a UA session.

- Session resume request: This interface is used by UC to resume a suspended UA session. Privilege check for UC is pre-requisite for resumption of a UA session (if the user is not the same as the one who suspend the session).
- Invitation request and invitation rejection: This interface is used by UC to invite other users to join the existing UA session, and this interface is also used to send rejection reason (if any).
- Monitoring registration request: This interface is used by UC to subscribe for getting notification for certain UC whenever this certain UC is online or this certain UC established a UA session.

6.2.2.2 UVE-2

This interface is exposed by VM and used by VMM. The functionalities supported on this interface are:

- Monitoring notification: This interface is used by VMM to notify VM about a monitoring request. In this notification VMM identifies the user who wishes to monitor an existing UA session. This will aid VM to consider the requesting user to be the authorized user for monitoring an existing session.
- Managing VM(s): This interface is used by VMM to deploy UA on VM, delete UA on VM and update UA on VM.
- Session suspend: This interface is used by VMM to suspend a monitored UA session.
- UA deployment: This interface is used by VMM to send available application and storefront identification to VM for application download/deployment.

6.2.2.3 UVE-3

This interface is exposed by VM and used by UC. The functionalities supported on this interface are:

- UVE Application Session request: After getting the selected VM in response to the initial request, UC will connect to the selected VM to establish an UVE application session, in which user actually starts using the application.
- Interaction Exchange: This interface is used for sending user interaction to VM.
- Terminal function initiation: This interface is used by VM to send a required terminal function (vibration, flash) to be executed/initiated to UC.
- Session Disconnect: This interface is also used to send session disconnect request to VM.

6.2.2.4 UVE-4

This interface is exposed by UC and used by VM to request the required Local Resources as per the requirements from UA.

6.2.2.5 UVE-5

This interface is exposed by VMM and used by VM. The functionalities supported on this interface are:

- This interface is used to notify VMM about the un-expected session disconnect. In the notification to VMM, VM identifies the session, user and UA related to the disconnected UA session.
- VM Data retrieve and update: This interface is used by VM to retrieve VM Data when the VM is selected to serve for the user. And it is also used by VM to update VM Data when user finishes consuming the VM or disconnected unexpectedly.

6.2.2.6 UVE 6

This interface is exposed by UVE Client and used by VMM.

- This interface is used to send notification about a Watchee to UVE Client. These notifications can be used to assist user to decide if he/she wants to start monitoring. The notification may relate to several actions like whenever the Watchee is online or has established its application session. These notifications are subject to user's pre-registration to receive these notifications.
- This interface is used by VMM to get the Watcher consent at runtime.
- This interface is also used by VMM to send suspend and resume notification to UC.
- Invitation Notification: This interface is used by VMM to notify UC about a session invitation.

6.2.2.7 UVE-7

This interface is exposed by VMM and used by existing storefronts to initiate a request for providing the application (as UA) to UVE infrastructure.

7. Technical Specification

7.1 Interfaces Descriptions

This section defines various interfaces used in UVE enabler. This includes defining messages exchanged on those interfaces with the parameters included.

7.1.1 UVE-1

This is the interface between UC and VMM. This interface includes the following operations.

7.1.1.1 UA Request

This operation allows requesting a particular UA. This involves sending a UARequest message to VMM and receiving UAResponse message in response.

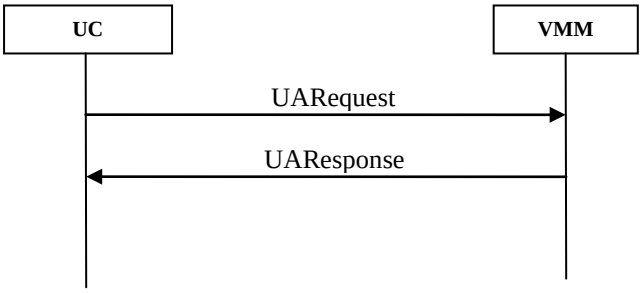

Figure 3: Flow of UARequest and its response

Message	Requirement	Direction
UARequest	Mandatory	UC → VMM
UAResponse	Mandatory	UC ← VMM

Table 1

7.1.1.1.1 UARequest

Name	Cardinality	XML Type	Data Type	Description
UARequest	1	E	String	Indicate the name of the message
AppID	1	E	String	A unique identification of a UA to be consumed.
UserID	0...1	E	String	A unique identification for the user.
TerminalCap	1	E	Structure	Structure containing terminal capabilities including display capabilities.
Reconnection Option	0…1	E	Boolean	A Boolean value stating whether No-Loss Reconnection is required or not. The default value is FALSE.
InputAllowed	0…1	E	Integer	If present, specifies the number of input devices allowed, by user, to connect with the requested UA.

Table 2

7.1.1.1.2 UAResponse

Name	Cardinality	XML Type	Data Type	Description
UAResponse	1	E	String	Indicate the name of the message
VMID	1	E	String	Indicates the identification information about the selected VM.
Status	1	E	Structure	Information about the status of the request

Table 3

7.1.1.2 Monitoring

This operation allows users to monitor an existing UA session.

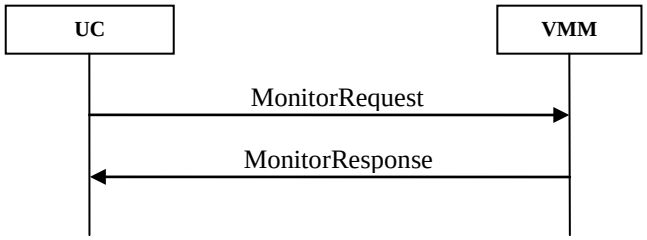

Figure 4: Flow of Monitor and its response

Message	Requirement	Direction
MonitorRequest	Mandatory	UC → VMM
MonitorResponse	Mandatory	UC ← VMM

Table 4

7.1.1.2.1 MonitorRequest

Name	Cardinality	XML Type	Data Type	Description
MonitorRequest	1	E	String	Indicate the name of the message
WatcheeID	1	E	String	The unique identification of the user whose UA session is to be monitored.
UserID	0...1	E	String	A unique identification for the user.
TerminalCap	1	E	Structure	Structure containing terminal capabilities including display capabilities.
AppID	0…1	E	String	UA identification.

Table 5

7.1.1.2.2 MonitorResponse

Name	Cardinality	XML Type	Data Type	Description
MonitorResponse	1	E	String	Indicate the name of the message
VMID	1	E	String	A unique identification for a VM to which UC shall connect to monitor the required session.
Status	0...1	E	Structure	Information about the status of the request.

Table 6

7.1.1.3 SessionSuspend

This operation allows users to suspend an existing UA session which he/she is monitoring.

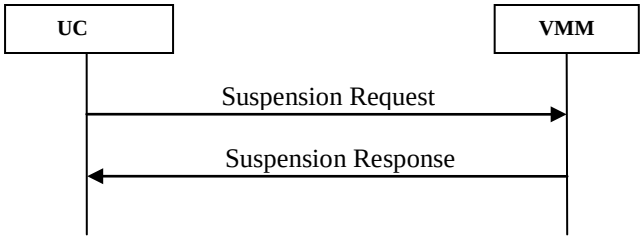

Figure 5: Figure of Suspension Request and Suspension Response

Message	Requirement	Direction
Suspension Request	Mandatory	UC → VMM
Suspension Response	Mandatory	UC ← VMM

Table 7

7.1.1.3.1 SuspensionRequest

Name	Cardinality	XML Type	Data Type	Description
Suspension Request	1	E	String	Indicate the name of the message
UserID	1	E	String	A unique identification for the user.
WatcheeID	1	E	String	The unique identification of the user whose UA session is to be suspended.
Cause	0...1	E	String	The reason to indicate why suspension is requested.
AppID	0…1	E	String	UA identification.

Table 8

7.1.1.3.2 SuspensionResponse

Name	Cardinality	XML Type	Data Type	Description
Suspension Response	1	E	String	Indicate the name of the message
Status	0...1	E	Structure	Information about the status of the request.

Table 9

7.1.1.4 SessionResumption

This operation allows users to resume an existing UA session.

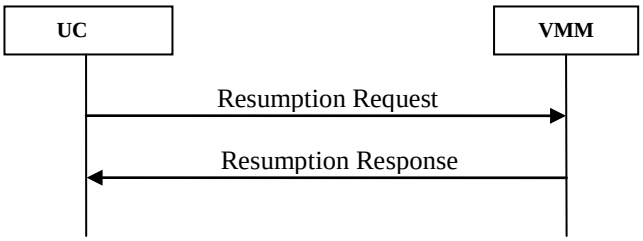

Figure 6: Figure of Resumption Request and Resumption Response

Message	Requirement	Direction
Resumption Request	Mandatory	UC → VMM
Resumption Response	Mandatory	UC ← VMM

Table 10

7.1.1.4.1 Resumption Request

Name	Cardinality	XML Type	Data Type	Description
ResumptoinRequest	1	E	String	Indicate the name of the message
UserID	1	E	String	A unique identification for the user.
WatcheeID	1	E	String	The unique identification of the user whose UA session is to be resumed.
Cause	0...1	E	String	The reason to indicate why resumption is requested.
AppID	0…1	E	String	UA identification.

Table 11

7.1.1.4.2 ResumptionResponse

Name	Cardinality	XML Type	Data Type	Description
Resumption Response	1	E	String	Indicate the name of the message
Status	0...1	E	Structure	Information about the status of the request.

Table 12

7.1.1.5 Monitoring Notification Registration

This operation allows UC to subscribe for getting notification for certain UC (Watchee) whenever this certain UC (Watchee) is online or this certain UC established a UA session.

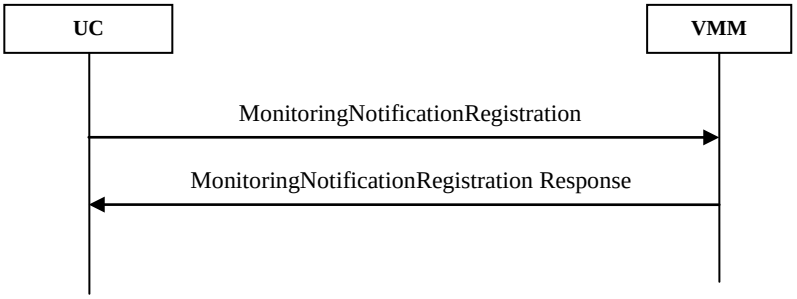

Figure 7: Flow of Monitoring Notification Registration Message Exchange

Message	Requirement	Direction
MonitoringNotificationRegistration	Mandatory	UC → VMM
MonitoringNotificationRegistrationResponse	Mandatory	UC ← VMM

Table 13

7.1.1.5.1 MonitoringNotificationRegistration

Name	Cardinality	XML Type	Data Type	Description
MonitoringNotificationRegistration	1	E		Indicate the name of the message
WatcheeID	1	E	String	The unique identification of the user for which monitoring notification are requested
UserID	0...1	E	String	A unique identification for the user.

Table 14

7.1.1.5.2 MonitoringNotificationRegistration Response

Name	Cardinality	XML Type	Data Type	Description
MonitoringNotificationRegistration Response	1	E		Indicate the name of the message
Status	1	E	Structure	Information about the status of the request.

Table 15

7.1.1.6 UAInvite request

This operation allows UC to invite other users to join its UA session. This involves sending a UAInviteRequest to VMM and receiving UAIniteResponse in response.

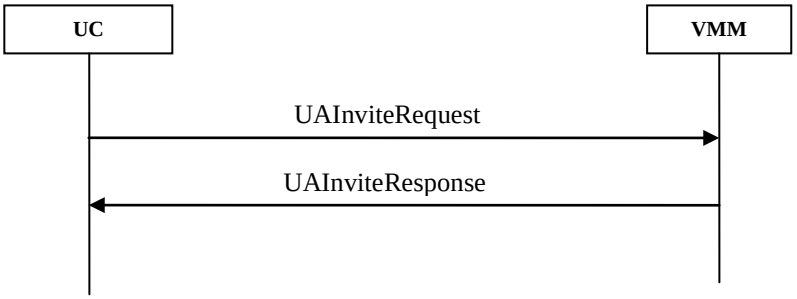

Figure 8: Flow of UAInvite Message Exchange

Message	Requirement	Direction
UAInviteRequest	Mandatory	UC → VMM
UAInviteResponse	Mandatory	UC ← VMM

Table 16

7.1.1.6.1 UAInviteRequest

Name	Cardinality	XML Type	Data Type	Description
UAInviteRequest	1	E		Indicate the name of the message
UserID	1	E	String	A unique identification for the user requested to consume the said application.
AppID	1	E	String	A unique identification of UA to be consumed.
InviteeID	1	E	String	A unique identification for the invitee user to be invited

Table 17

7.1.1.6.2 UAInviteResponse

Name	Cardinality	XML Type	Data Type	Description
UAInviteResponse	1	E		Indicate the name of the message
Status	1	E	Structure	Information about the status of the request.

Table 18

7.1.1.7 Reconnection

This operation allows UC to reconnect to an application session which was disrupted / disconnected unexpectedly. UC SHALL use the UARequest message to reconnect.

7.1.1.8 Runtime privilege request

This operation allows Wacthee to provide his/her consent as requested by VMM at runtime.

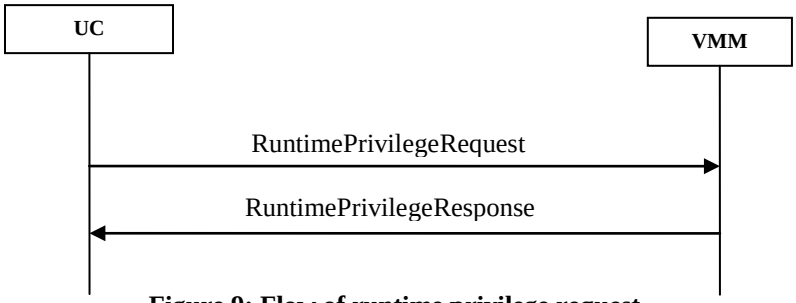

Figure 9: Flow of runtime privilege request

Message	Requirement	Direction
RuntimePrivilegeRequest	Mandatory	UC → VMM
RuntimePrivilegeResponse	Mandatory	UC ← VMM

Table 19

7.1.1.8.1 RuntimePrivilegeRequest

Name	Cardinality	XML Type	Data Type	Description
RuntimePrivilegeRequest	1	E		Indicate the name of the message
WatcherID	1	E	String	The unique identification of the Watcher
AppID	1	E	String	The identification of the UA with which the UC is connected.
Allow	1	E	Boolean	A boolean parameter providing Watchee decision on the request. Default is FALSE. If TRUE, imply a positive response from Watchee i.e. Watchee agrees to have Watcher monitoring his/her session.

Table 20

7.1.1.8.2 RuntimePrivilegeResponse

Name	Cardinality	XML Type	Data Type	Description
RuntimePrivilegeResponse	1	E		Indicate the name of the message
Status	1	E	Structure	Information about the status of the request.

Table 21

7.1.1.9 Session Join

This operation allows users to join the existing UA session.

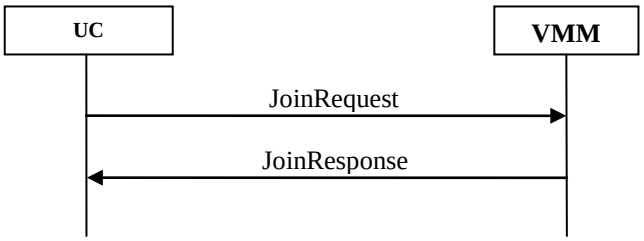

Figure 10: Flow of Join and its response

Message	Requirement	Direction
JoinRequest	Mandatory	UC → VMM
JoinResponse	Mandatory	UC ← VMM

Table 22

7.1.1.9.1 JoinRequest

Name	Cardinality	XML Type	Data Type	Description
JoinRequest	1	E	String	Indicate the name of the message
JoineeID	1	E	String	The unique identification of the user whose UA session is to be joined.
UserID	0...1	E	String	A unique identification for the user.
TerminalCap	1	E	Structure	Structure containing terminal capabilities including display capabilities.
AppID	1	E	String	A unique identification of a UA to be consumed.

Table 23

7.1.1.9.2 JoinResponse

Name	Cardinality	XML Type	Data Type	Description
JoinResponse	1	E	String	Indicate the name of the message
VMID	1	E	String	A unique identification for a VM to which UC shall connect to join the required session.
Status	0...1	E	Structure	Information about the status of the request.

Table 24

7.1.2 UVE-2

This is the interface between VM and VMM. This interface includes the following operations.

7.1.2.1 VMInitiateRequest

This operation allows VMM to initiate a selected VM. This VM will be contacted by UC after receiving UAResponse.

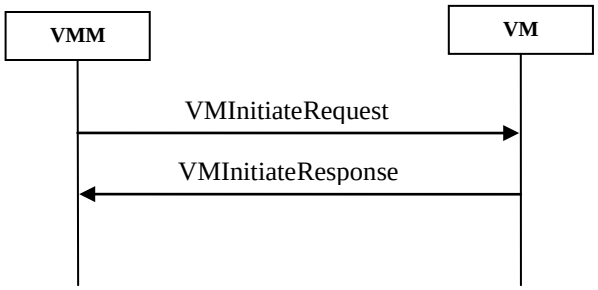

Figure 11: Flow of VMInitiateRequest and its response

Message	Requirement	Direction
VMInitiateRequest	Mandatory	VMM → VM
VMInitiateResponse	Mandatory	VMM ← VM

Table 25

7.1.2.1.1 VMInitiateRequest

Name	Cardinality	XML Type	Data Type	Description
VMInitiateRequest	1	E	String	Indicate the name of the message
AppID	1	E	String	A unique identification App to be consumed.
UserID	0...1	E	String	A unique identification for the user requested to consume the said application.
UIFURL	0…1	E	String	Location of the data centre in VMM where UIF is stored.

Table 26

7.1.2.1.2 VMInitiateResponse

Name	Cardinality	XML Type	Data Type	Description
VMInitiateResponse	1	E	String	Indicate the name of the message
Status	1	E	Structure	Information about the status of the request.

Table 27

7.1.2.2 UA Deployment

This operation allows VMM to notify selected VM about an available UA(s) which VM should get from 3-party and deploy.

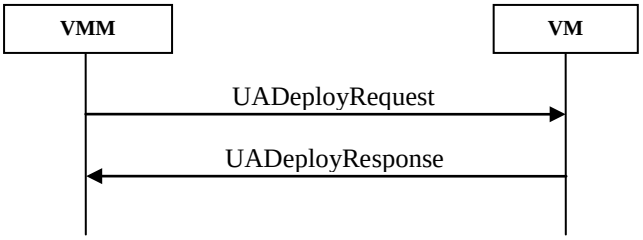

Figure 12: Flow of UADeployRequest and its response

Message	Requirement	Direction
UADeployRequest	Mandatory	VMM → VM
UADeployResponse	Mandatory	VMM ← VM

Table 28

7.1.2.2.1 UADeployRequest

Name	Cardinality	XML Type	Data Type	Description
UADeployRequest	1	E	String	Indicate the name of the message
Application	1…N	E	Structure	Information about application(s) available for download.
StoreID	1	E	String	Storefront contact point.

Table 29

7.1.2.2.2 UADeployResponse

Name	Cardinality	XML Type	Data Type	Description
UADeployResponse	1	E	String	Indicate the name of the message
Status	1	E	Structure	Information about the status of the request.

Table 30

7.1.2.3 Notifications

This operation allows sending notification to VM about a UA request and Monitoring request.

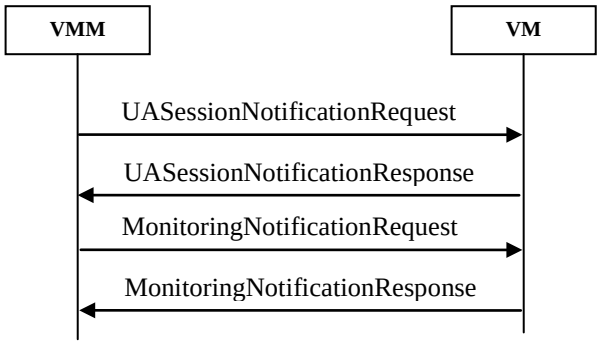

Figure 13: Flow of two Notification request and their response

Message	Requirement	Direction
UASessionNotificationRequest	Mandatory	VMM → VM
UASessionNotificationResponse	Mandatory	VMM ← VM
MonitoringNotificationRequest	Mandatory	VMM → VM
MonitoringNotificationResponse	Mandatory	VMM ← VM

Table 31

7.1.2.3.1 UASessionNotificationRequest

Name	Cardinality	XML Type	Data Type	Description
UASessionNotificationRequest	1	E	String	Indicate the name of the message
UserID	1	E	String	A unique identification for the requesting user.
APPId	1	E	String	A unique identification of the requested application.
InputAllowed	0…1	E	Integer	If present, specifies the number of input devices allowed, by user, to connect with the requested UA.

Table 32

7.1.2.3.2 UASessionNotificationResponse

Name	Cardinality	XML Type	Data Type	Description
UASessionNotificationResponse	1	E	String	Indicate the name of the message
Status	1	E	String	Information about the status of the request.

Table 33

7.1.2.3.3 MonitoringNotificationRequest

Name	Cardinality	XML Type	Data Type	Description
MonitorNotificationRequest	1	E	String	Indicate the name of the message
UserID	1	E	String	Watcher identification: Identification of an authorized user to connect to VM for monitoring purpose.
CanSuspend	0...1	E	Boolean	A Boolean value TRUE will state that the said Watcher has rights to suspend the monitored session. The default is FALSE.

Table 34

7.1.2.3.4 MonitoringNotificationResponse

Name	Cardinality	XML Type	Data Type	Description
MonitorNotificationResponse	1	E	String	Indicate the name of the message
Status	1	E	String	Information about the status of the request.

Table 35

7.1.3 UVE-3

This is the interface between UC and VM. This interface includes the following operations.

7.1.3.1 UA session setup

This operation allows UC and VM to negotiate a media session as per the rules and procedures defined in [Offer-Answer]. This media session will be a [RTP] session and will be used to deliver Output Stream from VM to UC. This involves sending a UASessionRequest to VM and receiving UASessionResponse in response.

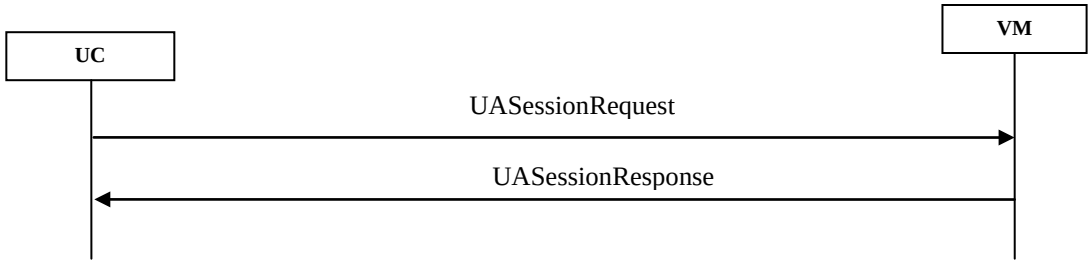

Figure 14: Flow of UASessionRequest and its response

Message	Requirement	Direction
UASessionRequest	Mandatory	UC → VM
UASessionResponse	Mandatory	UC ← VM

Table 36

7.1.3.1.1 UASessionRequest

Name	Cardinality	XML Type	Data Type	Description
UASessionRequest	1	E	String	Indicate the name of the message
UserID	1	E	String	A unique identification for the user.
APPId	1	E	String	A unique identification of the requested application.
InputDeviceID	0…1	E	String	The identification of the current input device connected to the terminal.
SDP	1	E	String	A set of parameters related to SDP offer (as described in [Offer-Answer]) facilitating the media session setup between UC and VM.

Table 37

7.1.3.1.2 UASessionResponse

Name	Cardinality	XML Type	Data Type	Description
UASessionResponse	1	E	String	Indicate the name of the message
Status	1	E	Structure	Information about the status of the request.
SDP	1	E	String	A set of parameters related to SDP answer (as described in [Offer-Answer]) facilitating the media session setup between UC and VM.

Table 38

7.1.3.2 User Interaction exchange

This operation allows UC to send different type of user interaction to VM in order to get them executed on the UA they are using.

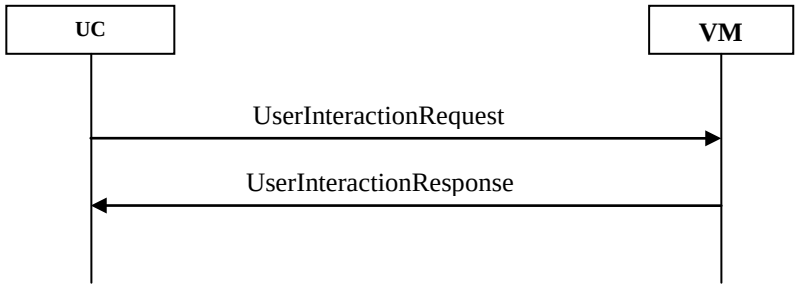

Figure 15: Flow of UserInteractionRequest and its response

Message	Requirement	Direction
UserInteractionRequest	Mandatory	UC → VM
UserInteractionResponse	Optional	UC ← VM

Table 39

7.1.3.2.1 UserInteractionRequest

Name	Cardinality	XML Type	Data Type	Description
UserInteraction Request	1	E		Indicate the name of the message
TouchInteractio n	0…1	E	Structure	Structure containing user's touch interaction on UA.
KeyInteraction	0...1	E	Structure	A structure containing keyboard related user interactions
AccelerometerI nteraction	0...1	E	Structure	A structure containing raw sensor data from accelerometer.
Mouse	0…1	E	Structure	Structure containing user's mouse interaction on UA.

Table 40

7.1.3.2.2 UserInteractionResponse

Name	Cardinality	XML Type	Data Type	Description
UserInteraction Response	1	E		Indicate the name of the message
Status	1	E	Structure	Information about the status of the request.

Table 41

7.1.3.3 UA session disconnect

This operation allows UC to disconnect a UA session.

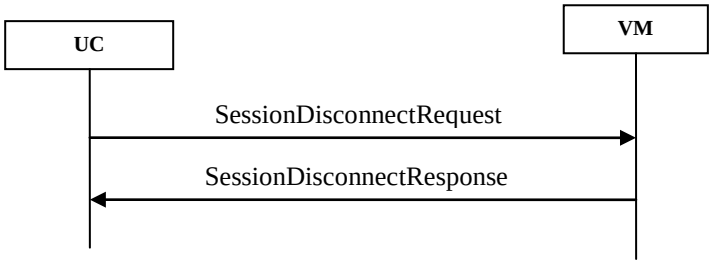

Figure 16: Flow of UASessionRequest and its response

Message	Requirement	Direction
SessionDisconnectRequest	Mandatory	UC → VM
SessionDisconnectResponse	Mandatory	UC ← VM

Table 42

7.1.3.3.1 SessionDisconnectRequest

Name	Cardinality	XML Type	Data Type	Description
SessionDisconn ectRequest	1	E		Indicate the name of the message
UserID	1	E	Structure	A unique identification for the user.
AppID	1	E	String	A unique identification of the UA which UC is using.

Table 43

7.1.3.3.2 SessionDisconnectResponse

Name	Cardinality	XML Type	Data Type	Description
SessionDisconnectResponse	1	E	String	Indicate the name of the message
Status	1	E	Structure	Information about the status of the request.

Table 44

7.1.3.4 UA ready notification

This operation allows VM to send a notification to all the UC(s) connected to the same UA stating that the number of UC(s) allowed connecting to this UA has reached. This will mean that UC can start interacting with the UA.

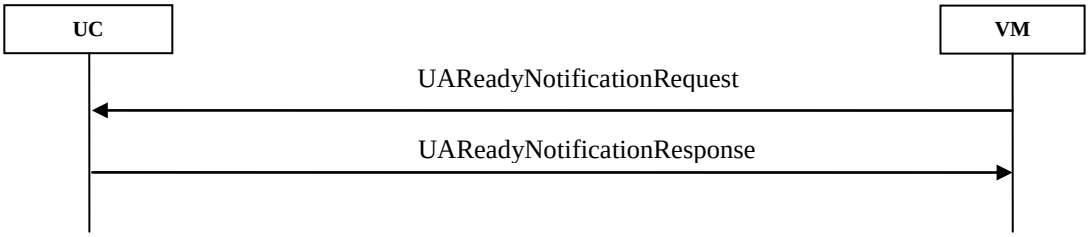

Figure 17: Flow of UAReadyNotificationRequest and its response

Message	Requirement	Direction
UAReadyNotificationRequest	Mandatory	UC → VM
UAReadyNotificationResponse	Mandatory	UC ← VM

Table 45

7.1.3.4.1 UAReadyNotificationRequest

Name	Cardinality	XML Type	Data Type	Description
UAReadyNotificationRequest	1	E	String	Indicate the name of the message
UAReady	1	E	Boolean	A Boolean attribute specifying whether the UA is ready to be consumed or not. If TRUE imply UA is ready. Default is FALSE.

Table 46

7.1.3.4.2 UAReadyNotificationResponse

Name	Cardinality	XML Type	Data Type	Description
UAReadyNotificationResponse	1	E	String	Indicate the name of the message
Status	1	E	Structure	Information about the status of the request.

Table 47

7.1.4 UVE-4

This is the interface between UC and VMM. This interface includes the following operations.

7.1.4.1 Local Resource Usage

This operation allows application to use local resource available to the terminal.

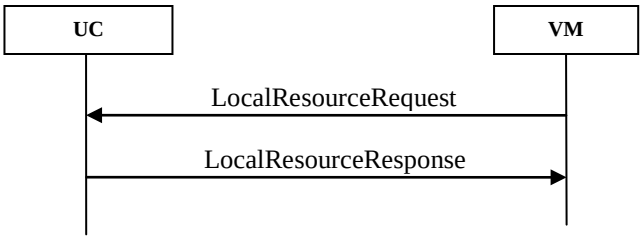

Figure 18: Flow of Local Resource Usage request and response

Message	Requirement	Direction
LocalResourceRequest	Mandatory	UC ← VMM
LocalResourceResponse	Mandatory	UC → VMM

Table 48

7.1.4.1.1 LocalResourceRequest

Name	Cardinality	XML Type	Data Type	Description
LocalResource Request	1	E	String	Indicate the name of the message
LRRequested	1	E	Structure	Structure identifying the requested local resource

Table 49

7.1.4.1.2 LocalResourceResponse

Name	Cardinality	XML Type	Data Type	Description
LocalResource Response	1	E	String	Indicate the name of the message
LRData	1	E	Structure	Structure containing the requested Local Resource data

Table 50

7.1.4.2 VM initiated terminal functions

This operation allows VM to request for execution of a particular function/interaction on the UC on behalf of UA. The function may include vibration, flash etc. The VMInitiatedFunctionRequest and VMInitiatedFunctionResponse are used for this purpose.

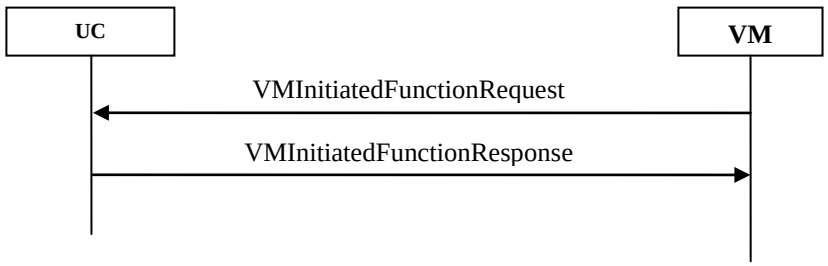

Figure 19: Flow of VMInitiatedFunctionRequest and its response

Message	Requirement	Direction
VMInitiatedFunctionRequest	Mandatory	UC ← VM
VMInitiatedFunctionRespons e	Mandatory	UC → VM

Table 51

7.1.4.2.1 VMInitiatedFunctionRequest

Name	Cardinality	XML Type	Data Type	Description
VMInitiatedFunctionRequest	1	E		Indicate the name of the message
FunctionRequested	1	E	ENUM	Structure containing information about the requested functionality. VM Will selects the appropriate value by intercepting the related function call by UA.

Table 52

7.1.4.2.2 VMInitiatedFunctionResponse

Name	Cardinality	XML Type	Data Type	Description
UASessionResponse	1	E	String	Indicate the name of the message
Status	1	E	Structure	Information about the status of the request.

Table 53

7.1.5 UVE-5

This is the interface between VMM and VM. This interface includes the following operations.

7.1.5.1 Exception Notification

This operation allows VM to notify VMM about an unexpected session disconnect.

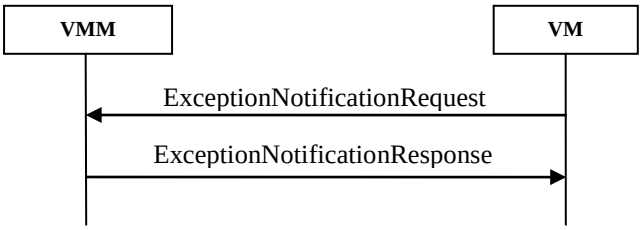

Figure 20: Flow of UARequest and its response

Message	Requirement	Direction
ExceptionNotificationRequest	Mandatory	VMM ← VM
ExceptionNotificationResponse	Mandatory	VMM → VM

Table 54

7.1.5.1.1 ExceptionNotificationRequest

Name	Cardinality	XML Type	Data Type	Description
ExceptionNotificationRequest	1	E	String	Indicate the name of the message
UserID	1	E	String	A unique identification of the UC which was connected to the VM before the exception occurred.
AppID	1	E	String	A unique identification of the UA which UC was using before the exception occurred.

Table 55

7.1.5.1.2 ExceptionNotificationResponse

Name	Cardinality	XML Type	Data Type	Description
ExceptionNotificationResponse	1	E	String	Indicate the name of the message
Status	0...1	E	Structure	Information about the status of the request.

Table 56

7.1.5.2 Disconnect Notification

This operation allows VM to notify VMM about a session disconnect.

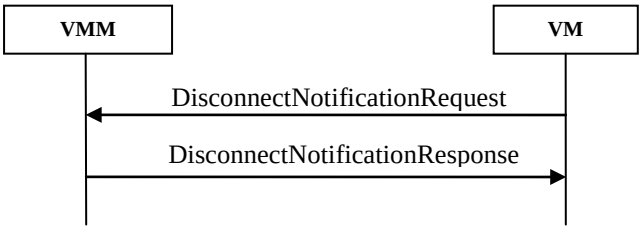

Figure 21: Flow of DisconectNotificationRequest and its response

Message	Requirement	Direction
DisconnectNotificationRequest	Mandatory	VMM ← VM
DisconnectNotificationResponse	Mandatory	VMM → VM

Table 57

7.1.5.2.1 DisconnectNotificationRequest

Name	Cardinality	XML Type	Data Type	Description
DisconnectNotificationRequest	1	E	String	Indicate the name of the message
UserID	1	E	String	A unique identification of the UC which was connected to the VM before the exception occurred.
AppID	1	E	String	A unique identification of the UA which UC was using.

Table 58

7.1.5.2.2 DisconnectNotificationResponse

Name	Cardinality	XML Type	Data Type	Description
ExceptionNotificationResponse	1	E	String	Indicate the name of the message
Status	0...1	E	Structure	Information about the status of the request.

Table 59

7.1.6 UVE-6

This is the interface between UC and VMM. This interface includes the following operations.

7.1.6.1 Monitoring Notification

This operation allows VMM to notify UC whenever the Watchee is online or has established its application session. These notifications are subject to user's pre-registration to receive these notifications.

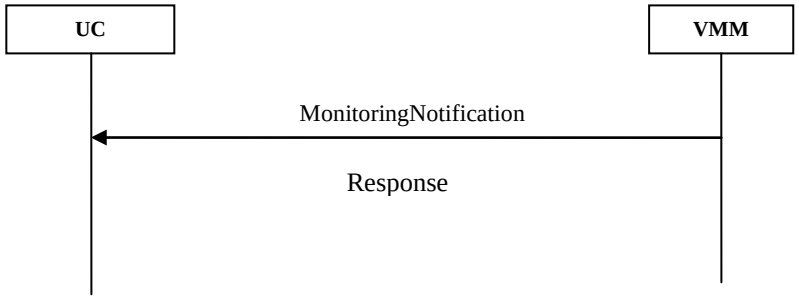

Figure 22: Flow of Monitoring Notification Registration Message Exchange

Message	Requirement	Direction
MonitoringNotification	Mandatory	UC ← VMM

Table 60

7.1.6.1.1 MonitoringNotification

Name	Cardinality	XML Type	Data Type	Description
MonitoringNotification	1	E		Indicate the name of the message
WatcheeID	1	E	String	The unique identification of the Watchee

Table 61

7.1.6.2 Runtime privilege request notification

This operation allows VMM to notify Watchee's about a monitoring request. Thereafter, Watchee can grant its consent at runtime.

Figure 23: Flow of runtime privilege request

Message	Requirement	Direction
RuntimePrivilegeRequestNotification	Mandatory	UC ← VMM

Table 62

7.1.6.2.1 RuntimePrivilegeRequestNotification

Name	Cardinality	XML Type	Data Type	Description
RuntimePrivilegeRequestNotification	1	E		Indicate the name of the message
WatcherID	1	E	String	The unique identification of the Watcher
AppID	1	E	String	The identification of the UA with which the UC is connected to and for which the monitoring request is received.

Table 63

7.1.6.3 Suspension Notification

This operation allows VMM to notify UC about a session suspension.

Figure 24: Flow of Suspension Notification Registration Message Exchange

Message	Requirement	Direction
SuspendNotificationRequest	Mandatory	UC ← VMM

Table 64

7.1.6.3.1 SuspendNotificationRequest

Name	Cardinality	XML Type	Data Type	Description
SuspendNotificationRequest	1	E		Indicate the name of the message
Reason	0...1	E	String	The reason to indicate why suspension is requested and probably by who. The parameter is based on "Cause" parameter of SuspensionRequest message.

Table 65

7.1.6.4 Resumption Notification

This operation allows VMM to notify UC about a session resumption

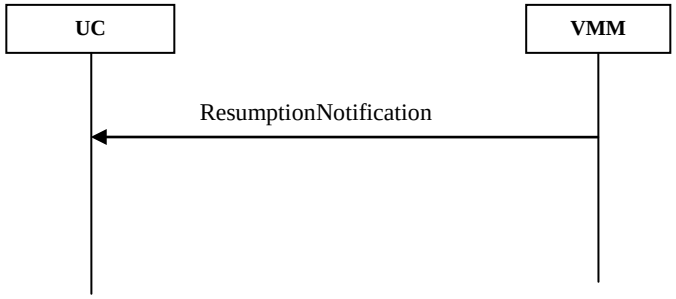

Figure 25: Flow of Resumption Notification Message Exchange

Message	Requirement	Direction
ResumptionNotification	Mandatory	UC ← VMM

Table 66

7.1.6.4.1 ResumptionNotification

Name	Cardinality	XML Type	Data Type	Description
ResumptionNotification	1	E		Indicate the name of the message
AppID	1	E	String	A unique identification of the UA UC should connect to consume.

Table 67

7.1.6.5 Invitation Notification

This operation allows VMM to notify UC about an invitation received. After receiving this notification UC decided whether to except the invitation and join the session.

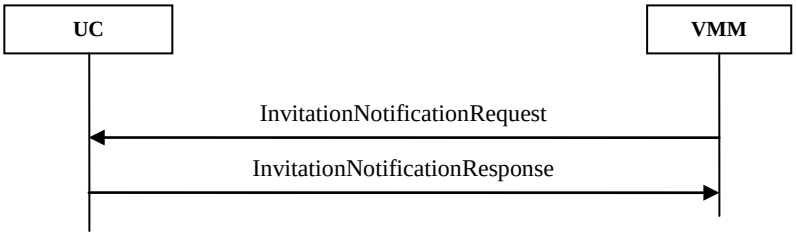

Figure 26: Flow of InvitationNotification request

Message	Requirement	Direction
InvitationNotificationRequest	Mandatory	UC ← VMM
InvitationNotificationResponse		

7.1.6.5.1 InvitationNotificationRequest

Name	Cardinality	XML Type	Data Type	Description
InvitationNotificationRequest	1	E		Indicate the name of the message
InviterID	1	E	String	The unique identification of the Inviting user
AppID	1	E	String	The identification of the UA with which the inviting user is connected to.

7.1.6.5.2 InvitationNotificationResponse

Name	Cardinality	XML Type	Data Type	Description
InvitationNotificationResponse	1	E		Indicate the name of the message
Status	1	E	Structure	Information about the status of the request
Reason	0..1	E	String	The reason parameter will convey the reason of rejection, if any.

7.1.7 UVE-7

This interface is between VMM and existing storefronts. This interface allows existing storefronts to initiate a request for providing the application (as UA) in UVE infrastructure. It supports the following operation.

7.1.7.1 Application Procurement

This operation allows existing storefronts to initiate a request for providing and deploying the application (as UA) in UVE infrastructure.

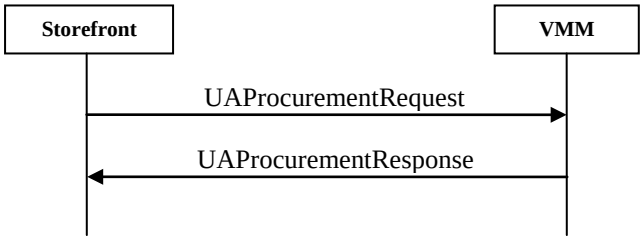

Figure 27: Flow of UAProcurementRequest and its response

Message	Requirement	Direction
UAProcurementRequest	Mandatory	Storefront → VMM
UAProcurementResponse	Mandatory	Storefront ← VMM

Table 68

7.1.7.1.1 UAProcurementRequest

Name	Cardinality	XML Type	Data Type	Description
UAProcurementRequest	1	E		Indicate the name of the message
ApplicationID	1…N	E	String	The identification of the application(s) available for download. The application identification will be completely storefront specific.
StoreID	1	E	String	The identifier of Storefront

Table 69

7.1.7.1.2 UAProcurementResponse

Name	Cardinality	XML Type	Data Type	Description
UAProcurementResponse	1	E		Indicate the name of the message
Status	1	E	Structure	Information about the status of the request.

Table 70

7.2 Entities Definitions

This section defines various entities of UVE enabler, which includes how those entities will behave to process different messages flowing on different interfaces defined.

7.2.1 UVE Client (UC)

7.2.1.1 UARequest

To send a UA request UC SHALL form a UARequest message.

7.2.1.2 MonitorResponse Handling

After receiving MonitorResponse, UC SHALL follow the following steps.

- UC SHALL connect to the VM identified by VMID using UASessionRequest message.
- UC SHALL stream the User Interaction towards VM as and when required.

7.2.1.3 UASessionResponse Handling

After receiving the UASessionResponse, UC can obtain and send the User Interaction towards VM as and when required.

7.2.1.4 LocalResourceRequest Handling

On receiving LocalResourceRequest UC shall follow the following steps

- UC SHALL collect the requested Local Resource data available to the terminal. UC MAY use native device APIs to get the required data.
- UC SHALL send the collected data to VM using LocalResourceResponse message.

7.2.1.5 VMInitiatedFunctionRequest Handling

On receiving VMInitiatedFunctionRequest message UC shall follow the following steps

- UC SHALL execute the function requested on the terminal. UC MAY use native device APIs for this.

7.2.1.6 ResumptionNotification Handling

After receiving ResumptionNotification message, UC SHALL send UARequest message to VMM. AppID in this message MUST be same as provided by VMM in ResumptionNotification.

7.2.1.7 UAReadyNotificationRequest Handling

After receiving UAReadyNotificationRequest on already established RTSP session, UC SHALL allow user to interact with the UA.

7.2.1.8 InvitationNotificationRequest Handling

On receiving InvitationNotificationRequest UC SHALL follow the following steps:

- Render the request to the user so that user can either accept or reject the invitation request.
- In case of rejection "Reason" parameter may contain the reason of rejection.
- If user accepts the invitation then UC SHALL use JoinRequest to join the session.

7.2.1.9 RuntimePrivilegeRequestNotification Handling

After receiving RuntimePrivilegeRequestNotification UC SHALL perform the following steps

- UC SHALL render the appropriate information to the user so that he/she can decide whether to grant the monitoring privilege.
- If user decides to grant the privilege, UC SHALL send RuntimePrivilegeRequest to VMM granting required privilege to Watcher.

7.2.2 Virtual Machine Manager (VMM)

7.2.2.1 MonitorRequest Handling

After receiving a MonitorRequest form UC, VMM SHALL follow the following steps

- VMM SHALL check if the UC has privilege to monitor the other UC's (identified as WatcheeID in the Monitor request) UA session. If the privilege check fails i.e. if UC doesn't have privilege to monitor Watchee's session then VMM MAY try to get Watchee consent at runtime using RuntimePrivilegeRequestNotification on UVE-6.
- VMM SHALL check terminal capabilities of the Watcher to monitor the Watchee's session.
    - If capabilities of Watcher and Watchee don't match, the monitoring request SHALL fail. The Status structure MAY include "incompatible capability" in StatusDes parameter.
- VMM SHALL identify the VM which is currently serving the user identified by WatcheeID provided in the Monitor request.
- VMM SHALL send the MonitoringNotificationRequest to the VM selected.

- VMM SHALL send identified VMID to the UC.

7.2.2.2 UARequest Handling

On receiving a UARequest form UC, VMM SHALL follow the following steps

- VMM SHALL authenticate and authorize UC.
- VMM SHALL find the appropriate VM by performing the following steps
    - Selects a VM(s) according to the AppID (provided in the UARequest) i.e. the VM(s) on which the required UA is installed or the VM(s) on which the required UA could be installed.
    - Then, selects a VM according to the TerminalCap (provided in the UARequest) i.e. the VM which is most suitable for the terminal as per the display capabilities are concerned.
- VMM SHALL send UASessionNotificationRequest to the VM selected.
- Depending on the UserID and AppID, VMM SHALL check for any previously saved VM Data to restore. If yes, the VMM SHALL restore the data in the selected VM. The exact restoring mechanism is considered out-of-scope of this enabler.
- VMM SHALL save the info provided in ReconnectionOption together with AppID and UserID.
- If the required UA is already installed on the selected VM, VMM SHALL then initiate the VM, otherwise, install the UA before initiating the VM.
- VMM SHALL send UAResponse message to UC containing VMID.

7.2.2.3 UAProcurmentRequest Handling

On receiving UAProcurementRequest VMM SHALL perform the following steps:

- VMM SHALL first select the appropriate VM as per the information provided by the Storefront in the request message.
- VMM SHALL then send a notification (UADeployRequest message) to the selected VM with the application ID and the point-of-contact for the storefront.

7.2.2.4 ExceptionNotificationRequest Handling

On receiving a ExceptionNotificationRequest form VM, VMM SHALL follow the following steps

- VMM SHALL identify the related UARequest based on UserID and AppID specified in the ExceptionNotificationRequest message.
- VMM SHALL save the VM Data/application data as per the ReconnectionOption specified in UARequest.
- VMM SHALL then release the VM resources.

7.2.2.5 SuspensionRequest Handling

After receiving a Suspension Request from UC, VMM SHALL follow the following steps

- VMM SHALL check if the UC has privilege to suspend the other UC's (identified as WatcheeID in the SuspensionRequest) UA session.
    - If UC does not have required privilege, VMM MAY try to get the UC's consent using RuntimePrivilegeRequestNotification on UVE 6.
- VMM SHALL identify the VM which is currently serving the user based on WatcheeID provided in the SuspensionRequest.
- VMM SHALL suspend Watchee session.
- VMM SHALL send SuspensionResponse to Watcher UC.

- VMM MAY send the SuspendNotificationRequest to Watchee UC.

7.2.2.6 JoinRequest Handling

After receiving a JoinRequest form UC, VMM SHALL follow the following steps

- VMM SHALL check terminal capabilities for probable incompatibility. In case incompatibilities exist, join request SHALL fail. The Status structure MAY include “incompatible capability” in StatusDes parameter.
- VMM SHALL identify the VM which is currently serving the user based on JoineeID provided in the join request.
- VMM SHALL send the VMID to the UC.
- UC will then use UASessionRequest to join the session with VM.

7.2.2.7 DisconnectNotificationRequest Handling

After receiving a DisconnectNotificationRequest VMM SHALL follow the following steps

- VMM SHALL release the VM sending the DisconnectNotification response.

7.2.2.8 UAInviteRequest Handling

After receiving a UAInvite Request from UC, VMM SHALL follow the following steps:

- VMM SHALL invite the UC identified by InviteeID using another message on UVE-6 (the push interface).
- VMM SHALL send UAInvite Response to the inviter UC.

7.2.2.9 Resumption Request Handling

After receiving a ResumptionRequest from UC, VMM SHALL follow the following steps

- Depending on the information provided in the request VMM SHALL reload the VM Data in a VM.
- VMM SHALL then send ResumptionNotification to UC, identified as WatcheeID, on UVE 6.

7.2.2.10 MonitoringNotificationRegistrationRequest Handling

After receiving MonitoringNotificationRegistrationRequest VMM SHALL send MonitoringNotification whenever the UC, identified as WatcheeID, establishes its UA session.

7.2.3 Virtual Machine (VM)

7.2.3.1 UASessionRequest Handling

After receiving the UASessionRequest VM SHALL follow the following steps

- VM SHALL authorize the requesting user using the information it received in UASessionNotification message.
- VM SHALL send a response using UASessionResponse message which shall establish an RTP session between UC and VM.
- If the InputDeviceID parameter is present then:
    - VM SHALL check the uniqueness of the InputDeviceID(s) previously received from the same user for the current session. If the InputDeviceID present in the request is not unique then VM SHALL reject the request with appropriate reason in the StatusDes parameter present in the response.
    - VM SHALL select an ideal (next available) Virtual Input Module and map it with the UserID and InputDeviceID received in the request.
    - VMM SHALL check whether the number of correct InputDeviceID(s) received for this UA has reached the number specified in InputAllowed parameter initially received for this UA;

    - If it is not, VM SHALL NOT accept any User Interaction from UC, if any, on the session established. Further steps here should be ignored.
- If it is, VM SHALL send UAReadyNotification message to all the UC(s) connected to this UA, at present.
- VM SHALL form a Output Stream, as specified in section 7.2.3.4, and start streaming in the session just established
- VM SHALL be ready to receive User Interaction from UC.

7.2.3.2 UASessionNotification Handling

On receiving the UASessionNotification request VM SHALL take the following steps:

- VMM SHALL save the UserID specified in the request among the authorize users who can connect to VM. This list will be used to authorize users on receiving UASessionRequest.
- VM SHALL create as many Virtual Input Modules as specified in the InputAllowed parameter.

7.2.3.3 UADeployRequestHandling

On receiving UADeployRequest VM SHALL download the application specified by VMM from the storefront. The download process should be storefront specific. This specification does not mandate any download process. However, this specification provides mechanism for downloading application from TAS [TAS] enabled storefront in Appendix E.

7.2.3.4 Output Stream provisioning

VM is responsible to form Output Stream and deliver it to UC on the RTP session established using UASessionRequest and its response. VM SHALL take the following steps

- VM SHALL capture the UA graphics. Appendix E demonstrates the method to capture graphics on android and Windows system.
- VM SHALL encode the captured graphics into a video stream (e.g. H.264)
- VM SHALL then send encoded video stream on RTP session using the procedures and guidelines defined in [AV-RTP].

7.2.3.5 LocalResourceRequest

To send a LocalResourceRequest VM SHALL perform the following steps.VM SHALL detect the type of Local Resource being accessed by UA at run time.

- VM SHALL identify the Device API being used by UA and map it with the corresponding API supported by the UC (based on Device OS being used at the terminal). The mapping will include API name mapping and API parameter mapping, if any.
- VM SHALL then form a LocalResourceRequest and send it to UC.

7.2.3.6 LocalResourceResponse handling

After receiving LocalResourceResponse message VM SHALL perform the following steps.

- VM SHALL check if the OS on UC is same with OS on VM. If they are not same VM SHALL format the data received in the response into the format compatible with VM's guest-OS.
- VM SHALL then provide the data to the requesting UA as expected.

7.2.3.7 MonitorNotificationRequest Handling

On receiving the MonitorNotificationRequest VM SHALL take the following steps:

- VMM SHALL save the UserID specified in the request among the authorize users who can connect to VM. This list will be used to authorize users on receiving UASessionRequest.

7.2.3.8 SessionDisconnectRequest Handling

On receiving SessionDisconenctRequest VM SHALL send the DisconnectNotificationRequest to VMM.

7.3 Commonly Used Structures

7.3.1 Status

Name	Cardinality	XML Type	Data Type	Description
StatusCode	1	E	String	Indicate status code
StatusDes	0...1	E	String	A related description, if any.

Table 71

7.3.2 VMID

Name	Cardinality	XML Type	Data Type	Description
IPAddress	0…1	E	String	IP address of the VM identified must be provided, if VM is deployed in local network
Port	0…1	E	String	The specific port on the VM must be provided, if VM is deployed in local network
URL	0…1	E	String	The URL on which the VM can be accessed must be provided, if VM is not deployed in local network

Table 72

7.3.3 TerminalCap

Name	Cardinality	XML Type	Data Type	Description
ScreenSize	0..1	E	Integer	The diagonal width of the display screen in mm
Resolution	0…1	E	Integer	Supported different resolution of the screen in terms of pixels
AV_Decoding	1…N	E	String	Audio/Video decoding format supported by the UC. Such as: g.721,mp3
Device-OS	1	E	String	Device OS used at terminal.

Table 73

7.3.4 TouchInteraction Structure

Name	Cardinality	XML Type	Data Type	Description
SingleTouch	0…1	E	Structure	Structure containing user's single touch interaction on UA.
MultiTouch	0…1	E	Structure	Structure containing user's multi-touch interaction on UA.

Table 74

7.3.5 SingleTouch Structure

Name	Cardinality	XML Type	Data Type	Description
tX	1	E	int	X coordinate
tY	1	E	int	Y coordinate
State	1	E	Enum	Enumeration with two values: UP and DOWN

Table 75

7.3.6 State Enumeration

Name	Description
UP	Identify the up state of touch/click
Down	Identify the down state of touch/click

Table 76

7.3.7 MultiTouch Structure

Name	Cardinality	XML Type	Data Type	Description
FirstTouch	1…N	E	SingleTouch type	SingleTouch structure representing first touch of multi touch interaction. In case of more than 1 occurrence first and last occurrences represent down and up action respectively.
SecondTouch	1…N	E	SingleTouch type	SingleTouch structure representing second touch of multi touch interaction. In case of more than 1 occurrence first and last occurrences represent down and up action respectively.

Table 77

7.3.8 KeyInteraction Structure

Name	Cardinality	XML Type	Data Type	Description
Keyvalue	0…1	E	String	A hexadecimal key code value of the key pressed.
State	0…1	E	Structure	Enumeration with two values: UP and DOWN

Table 78

7.3.9 AccelerometerInteraction Structure

Name	Cardinality	XML Type	Data Type	Description
xMotion	1	E	String	Device motion in the x direction
yMotion	1	E	String	Device motion in the y direction
zMotion	1	E	String	Device motion in the z direction

Table 79

7.3.10 Mouse Structure

Name	Cardinality	XML Type	Data Type	Description
cX	1	E	int	X coordinate
cY	1	E	int	Y coordinate
State	1	E	Enum	Enumeration with two values: UP and DOWN
ButtonValue	1	E	Enum	Enumeration specifying type of three mouse buttons: LEFT, RIGHT and MIDDLE

Table 80

7.3.11 ButtonValue Enumeration

Name	Description
LEFT	Left mouse button
RIGHT	Right mouse button
MIDDLE	Middle mouse button

Table 81

7.3.12 LRRequested Structure

Name	Cardinality	XML Type	Data Type	Description
LR-GPS	0…1	E	Structure	Structure specifying the Local Resource usage request for GPS data.

Table 82

7.3.13 LR-GPS Structure

Name	Cardinality	XML Type	Data Type	Description
APIName	1	E	String	API name used. This will help UC to invoke the appropriate native device API.
APIParam	0…1	E	Structure	The structure containing any parameter's name-value pair to be used with the API.

Table 83

7.3.14 LRData Structure

Name	Cardinality	XML Type	Data Type	Description
GPS-Data	0…1	E	Structure	Structure providing GPS data.

Table 84

7.3.15 GPS-Data Structure

Name	Cardinality	XML Type	Data Type	Description
Latitude	1	E	Double	Latitude
Longitude	1	E	Double	Longitude
Bearing	0…1	E	Float	Heading, in degrees. A heading of zero is true north
Speed	0…1	E	Float	Returns the speed of the device over ground in meters/second.
Altitude	0…1	E	Float	Altitude

Table 85

7.3.16 FunctionRequested Enumeration

Name	Description
VIBRATION	Vibrations function of the device.
FLASHLIGHT	Flash light of the camera.

Table 86

7.3.17 Application Structure

Name	Cardinality	XML Type	Data Type	Description
ID	1	Element	String	Unique Identification of the application in Storefront infrastructure.
Description	0…1	Element	String	Any related description.

Table 87

7.4 Protocol Bindings

7.4.1 HTTP Bindings

7.4.1.1 General

UC SHALL support Hypertext Transfer Protocol version 1.1 (HTTP1.1 [RFC2616]) for UVE 1.

VMM SHALL support Hypertext Transfer Protocol version 1.1 (HTTP1.1 [RFC2616]) for UVE 1, UVE 2, UVE 5 and UVE 7 interfaces.

VM SHALL support Hypertext Transfer Protocol version 1.1 (HTTP1.1 [RFC2616]) for UVE 2, UVE 5 and UVE 4 interfaces.

7.4.1.1.1 HTTP Method

All the request messages SHALL be sending as HTTP POST method requests.

The following optional Headers may be included in the request messages.

- the receiver's address in the request line
- the Host request-header set to the hostname or IP address of the receiver.
- the User-Agent request-header set to identify the host device (e.g. "vendor-model/version"), and the name and version of the sender as user agent initiating the request.
- the Accept request-header with value "application/xml, multipart/mixed"
- the Accept-Encoding request-header with value per the supported HTTP compression encodings, i.e. deflate and / or gzip
- the Accept-Language request-header with value per the supported HTTP supported languages(e.g. en, *)
- the Accept-MsgSize is the maximum message size that terminal can handle.
- the Content-Length entity-header set to the length of the entity-body
- the Content-Type entity-header with value "application/xml", "multipart/mixed" or "multipart/form-data" as applicable
- the Interface message(s) as message-body

If any of these headers are not present in the response to the request, the receiver SHALL assume their default values.

All the response messages SHALL be sending as response to the corresponding receiver's request as specified by the HTTP 1.1 including:

- Status-Line header reflects the outcome of the HTTP POST request
- the ETag entity-header set to a unique value within the scope of the receiver.
- the Content-Encoding entity-header set to the type of HTTP compression applied, if any
- the Content-Length entity-header set to the length of the entity-body
- the Content-Type entity-header with value "application/xml or "multipart/mixed", as applicable
- the Interface message(s) as message-body, if the transaction is successful

7.4.1.1.2 HTTP Authentication

The UC, VMM and VM SHALL support HTTP Digest Authentication mechanisms (HTTP1.1 [RFC2617]) on UVE 1, UVE 3, UVE 4 and UVE 7 interfaces.

7.4.1.2 UVE 1 Interface Message Bindings

The table below gives an overview of how UVE 1 interface messages are bound to the HTTP based protocol stack.

Message	UC ←→ VMM	HTTP Method
UARequest	→	HTTP POST
UAResponse	←	HTTP Response (including 200 OK of the underlying method)
MonitorRequest	→	HTTP POST
MonitorResponse	←	HTTP Response (including 200 OK of the underlying method)
SuspensionRequest	→	HTTP POST
SespensionResponse	←	HTTP Response (including 200 OK of the underlying method)
Resumption Request	→	HTTP POST
Resumption Response	←	HTTP Response (including 200 OK of the underlying method)
MonitoringNotificationRegistration	→	HTTP POST
MonitoringNotificationRegistrationResponse	←	HTTP Response (including 200 OK of the underlying method)
UAInviteRequest	→	HTTP POST
UAInviteResponse	←	HTTP Response (including 200 OK of the underlying method)
RuntimePrivilegeRequest	→	HTTP POST
RuntimePrivilegeResponse	←	HTTP Response (including 200 OK of the underlying method)
JoinRequest	→	HTTP POST
JoinResponse	←	HTTP Response (including 200 OK of the underlying method)

Table 88: UVE 1 Interface Message Bindings When Uses HTTP Based Stack

7.4.1.3 UVE 2 Interface Message Bindings

The table below gives an overview of how UVE 2 interface messages are bound to the HTTP based protocol stack.

Message	VMM ←→ VM	HTTP Method
VMInitiateRequest	→	HTTP POST
VMInitiateResponse	←	HTTP Response (including 200 OK of the underlying method)
UADeployRequest	→	HTTP POST
UADeployResponse	←	HTTP Response (including 200 OK of the underlying method)
UASessionNotificationRequest	→	HTTP POST
UASessionNotificationResponse	←	HTTP Response (including 200 OK of the

		underlying method)
MonitoringNotificationRequest	→	HTTP POST
MonitoringNotificationResponse	←	HTTP Response (including 200 OK of the underlying method)

Table 89: UVE 2 Interface Message Bindings When Uses HTTP Based Stack

7.4.1.4 UVE 4 Interface Message Bindings

The table below gives an overview of how UVE 4 interface messages are bound to the HTTP based protocol stack.

Message	VM ←→ UC	HTTP Method
LocalResourceRequest	→	HTTP POST
LocalResourceResponse	←	HTTP Response (including 200 OK of the underlying method)

Table 90: UVE 4 Interface Message Bindings When Uses HTTP Based Stack

7.4.1.5 UVE 5 Interface Message Bindings

The table below gives an overview of how UVE 5 interface messages are bound to the HTTP based protocol stack.

Message	VMM ←→ VM	HTTP Method
ExceptionNotificationRequest	→	HTTP POST
ExceptionNotificationResponse	←	HTTP Response (including 200 OK of the underlying method)
DisconnectNotificationRequest	→	HTTP POST
DisconnectNotificationResponse	←	HTTP Response (including 200 OK of the underlying method)

Table 91: UVE 5 Interface Message Bindings When Uses HTTP Based Stack

7.4.1.6 UVE 7 Interface Message Bindings

The table below gives an overview of how UVE 7 interface messages are bound to the HTTP based protocol stack.

Message	Storefront ←→ VMM	HTTP Method
UAProcurementRequest	→	HTTP POST
UAProcurementResponse	←	HTTP Response (including 200 OK of the underlying method)

Table 92: UVE 7 Interface Message Bindings When Uses HTTP Based Stack

7.4.2 RTSP Bindings

7.4.2.1 General

UC and VM SHALL support Real Time Streaming Protocol ([RTSP]) for UVE 3 interface.

7.4.2.2 UVE 3 Interface bindings

Message	UC ←→ VMM	HTTP Method
UASessionSetupRequest	→	RTSP DESCRIBE
UASessionSetupResponse	←	RTSP Response (including 200 OK of the underlying method)
UserInteractionRequest	→	RTSP DESCRIBE
UserInteractionResponse	←	RTSP Response (including 200 OK of the underlying method)
VMInitiatedFunctionRequest	→	RTSP DESCRIBE
VMInitiatedFunctionResponse	←	RTSP Response (including 200 OK of the underlying method)
SessionDisconnectRequest	→	RTSP DESCRIBE
SessionDisconnectResponse	←	RTSP Response (including 200 OK of the underlying method)
UAReadyNotificationRequest	→	RTSP DESCRIBE
UAReadyNotificationResponse	←	RTSP Response (including 200 OK of the underlying method)

Table 93

7.4.3 OMA PUSH Bindings

The VMM SHALL support either the Push Access Protocol (PAP) [PUSH-PAP] or the Push-OTA protocol [PUSH-OTA] for point-to-point delivery of the following messages over the UVE 6 interface:

- UVE 6 MonitoringNotification

Whether to use PAP or Push-OTA is a deployment choice. Among other advantages, use of PAP prevents the VMM from needing to implement most of the basic functions of OMA Push Proxy Gateway (PPG), e.g. the various Push-OTA protocols and target client context/capabilities awareness (as needed to select the appropriate transport bearer and protocol). Conversely, for high-volume services limited to Push delivery over a single specific bearer (e.g. WAP1 Push over SMS), use of Push-OTA is fairly simple and avoids dependency upon a Push Proxy Gateway (PPG).

UVE messages may not be directly deliverable over connectionless bearers such as WAP Push over SMS (limited to about 512 bytes) or SIP Push via the SIP MESSAGE method (limited to about 1300 bytes), as compared to WAP Push over HTTP or SIP Push via the INVITE/MSRP method (both of which support essentially unlimited content size). The alternative to direct delivery is “indirect delivery”, which involves the delivery of a Push notification message carrying a content location URL, from which the client retrieves the response.

When using PAP for direct UVE message delivery, the VMM SHALL submit the UVE messages using the MIME content type “application/xml”, and MAY support various target client address schemes, e.g. PLMN, USER, SIP URI, IP address, etc.

When directly delivering UVE messages via Push-OTA, the UVE SHALL send the UVE message (headers and message body) encapsulated into a message/vnd.oma.push media type as described in [PUSH-MSG].

To deliver UVE 6 messages indirectly, the VMM SHALL use the ServiceLoading [PUSH-SL] content type, via either PAP or Push-OTA as applicable, and include a URL from which the UC can retrieve the actual UVE 6 message. US SHALL support indirect delivery of UVE messages, triggered by reception of ServiceLoading notifications.

For UVE Push messages, the VMM SHALL include the Push Application ID header "X-Wap-Application-Id: x-oma-application:uve.ua". Push Clients in UVE supporting terminals SHALL support routing of Push messages with the Push Application ID header "X-Wap-Application-Id: x-oma-application:uve.ua" to the UC. If there is no Push Client in the device, the UC SHALL implements the necessary Push Client functions for the supported Push-OTA protocol variants per [PUSH-OTA].

7.4.3.1 UVE 6 Interface Message Bindings

The table below gives an overview of how UVE 6 interface messages are bound to the OMA Push-OTA or PAP protocol.

Message	UC ←→ VMM	Push-OTA Binding	PAP Binding
UVE 6 MonitoringNotification	←	Push Message (MonitoringNotification)	Push Message (MonitoringNotification)
OR			
UVE 6: MonitoringNotification URL Notification	←	ServiceLoading (MonitoringNotification URL)	ServiceLoading (MonitoringNotification URL)
After get the URL notification, UC can fetch the MonitoringNotification using the HTTP GET.			

Table 94: UVE 6 Monitoring notification Operation Bindings to OMA Push

8. Release Information

8.1 Supporting File Document Listing

Doc Ref	Permanent Document Reference	Description
Supporting File		
[UVE_DTD]	OMA-SUP-UVE_V1.0_XSD-V1_0-20151201-A	DTD for the messages and included elements of the UVE protocols. Working file in DTD directory: file: foo_msgs-v1_2.dtd path: http://www.openmobilealliance.org/tech/dtd/

Table 95: Listing of Supporting Documents in FOO Release

8.2 OMNA Considerations

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
OMA-ER-UVE-V1_0-20151201-A	01 Dec 2015	Status changed to Approved by TP TP Ref # OMA-TP-2015-0204-INP_UVE_V1_0_ERP_for_final_Approval

Appendix B. Use Cases (Informative)

Refer to [UVE-RD] for use cases.

Appendix C. Static Conformance Requirements (Normative)

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

C.1 SCR for UVE Client

Item	Function	Reference	Requirement
UVE-C-001-M	UARequest delivery on UVE-1.	7.1.1.1	
UVE-C-002-M	MonitorRequest delivery on UVE-1	7.1.1.2	
UVE-C-003-M	Suspending and resuming UA session using SuspensionRequest and ResumptionRequest respectively on UVE 1	7.1.1.3, 7.1.1.4	
UVE-C-003-O	MonitoringNotificationRegistration request delivery on UVE 1	7.1.1.5	
UVE-C-003-M	Inviting other user and joining existing session, on invitation, using UAInviteRequest and JoinRequest respectively on UVE 1.	7.1.1.6, 7.1.1.9	
UVE-C-003-M	RuntimePrivilegeRequest delivery on UVE 1	7.1.1.8	
UVE-C-003-M	Delivering SessionDisconnectRequest on UVE-3	7.1.3.4	

C.2 SCR for VM

Item	Function	Reference	Requirement
UVE-VM-001-M	Receiving the UASessionRequest from UC and establish an RTP session. Create an Output Stream and start streaming in the session just established.	7.2.3.1, 7.2.3.4	
UVE-VM-001-O	Receiving the UADeployRequest and retrieve the application from the appstore provided in the request.	7.2.3.3	
UVE-VM-001-M	Delivering LocalResourceRequest on UVE 4 to get the Local Resource data requested by the UA.	7.1.4.1, 7.2.3.5, 7.2.3.6,	
UVE-VM-001-M	Delivering VMInitiatedTerminalFun	7.1.4.2	

Item	Function	Reference	Requirement
	ctionRequest on UVE 4 to get the local function on the terminal as requested by UA.		
UVE-VM-001-M	Receiving SessionDisconnectRequest and sending DisconnectNotificationRequest to VMM on UVE 5.	7.2.3.7	

C.3 SCR for VMM

Item	Function	Reference	Requirement
UVE-VMM-001-M	Accepting UARequest and select an appropriate VM to initiate the VM with the requested UA.	7.1.1.1, 7.1.2.2	
UVE-VMM-001-O	Receiving the UAProcurmentRequest from other storefront, selecting the appropriate VM and notifying VM, using UADeployRequest, about the application(s) which it should get form the storefront.	7.1.7.1, 7.1.2.2, 7.2.2.3, 7.2.3.3	
UVE-VMM-001-M	Sending UA Session and Monitoring notification to VM using UASessionNotification and MonitoringNotification respectively.	7.1.2.3	
UVE-VMM-001-M	Receiving the monitoring request and checking the Watcher privilege to monitor the Watchee session.	7.2.2.1	
UVE-VMM-001-M	Releasing VM resources after getting DisconnectNotificationRequest form VM.	7.2.2.7	

Appendix D. Architectural Flows

D.1 UA Request

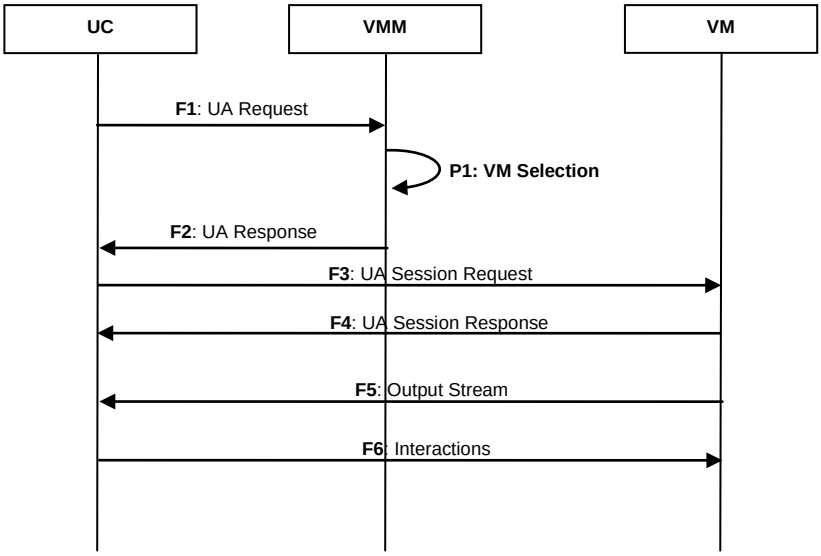

Figure 28

- **F1:** UC sends an initial request, containing the identification of the required UA, user preference (e.g. bandwidth, resolution) and terminal capabilities to VMM.
- **P1:** VMM selects the appropriate VM considering requested UA, user preference (e.g. bandwidth, resolution) and terminal capabilities. If the required UA is pre-installed on the selected VM, VMM will initiate the UA with the selected VM. Otherwise, VMM will install the required UA on the selected VM before initiating it.
- **F2:** VMM sends a response, containing the selected VM identification, to UC.
- **F3:** UC sends a connection request to the specified (in F2) VM.
- **F4:** VM accepts the request and send a response. The UA session is then established between UC and VM.
- **F5 & F6:** Once the UA session is established the interaction command and Output Stream will be exchanged between UC and VM.

D.2 Monitoring

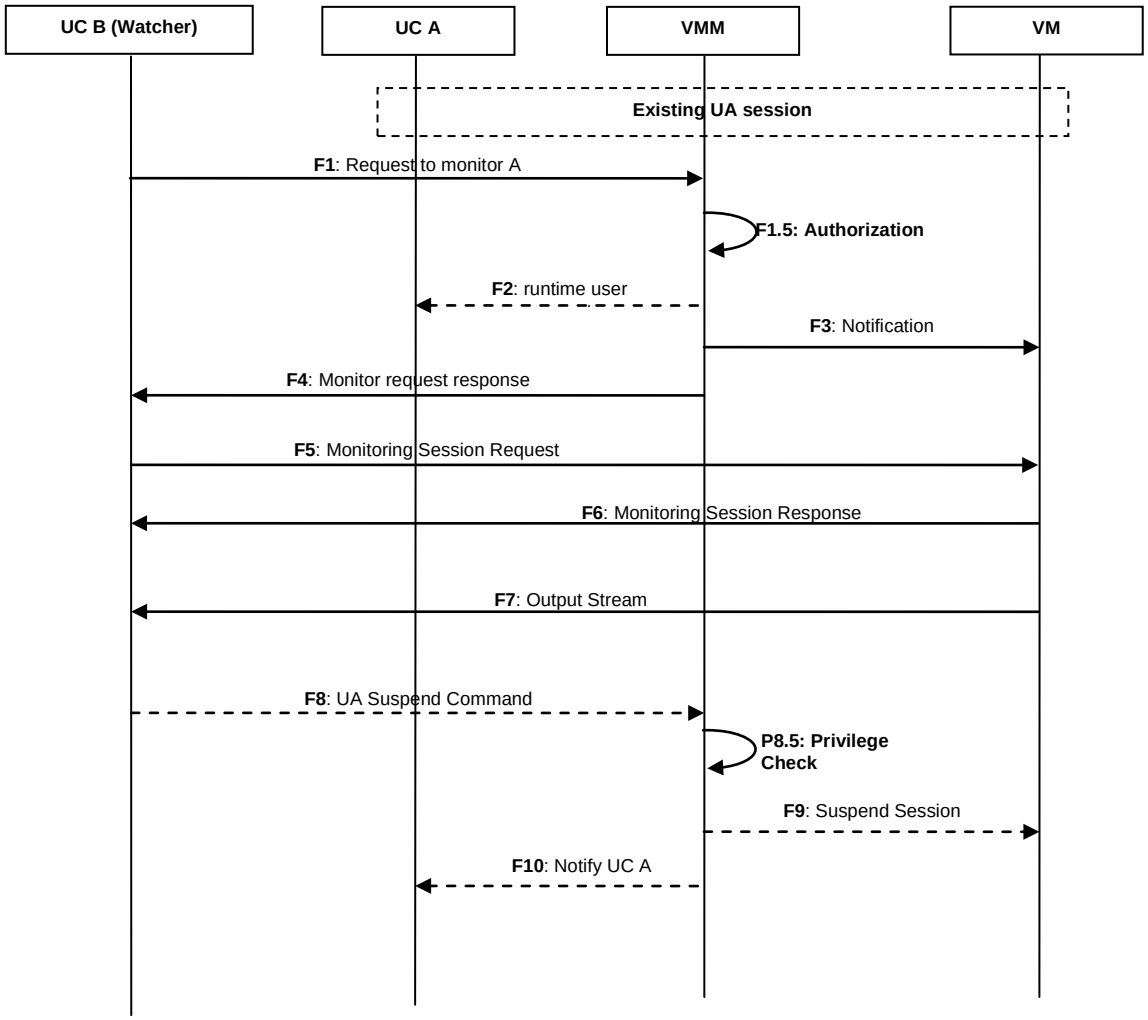

Figure 29

- Existing UVE application session in progress.
- **F1:** UC B sends a monitoring request to VMM to monitor an existing UVE session involving UC A. This request will contain UC A identification and UC B's terminal capabilities information
- **P1.5:** VMM will find the VM who is at present serving UC A. VMM will check if UC B is authorized to monitor UC A. It may involve checking A's Watcher List to see if UC B is present in the list. And, VMM will also check if UC B has compatible terminal capabilities to monitor UC A. If UC B has no compatible terminal capabilities to monitor UC A, the monitoring request fails and following steps are neglected.
- **F2:** In case UC B is not among the Watcher for UC A. VMM will try to get explicit user consent, at runtime, form UC A.
- **F3**: VMM notify VM, involved in the requested session, about the monitoring request. This is to make VM understand that an authorised user (UC B) will be connecting to monitor the existing session.
- **F4**: VMM sends a response to the monitoring request with the VM identification.
- **F5**: UC B sends a request to VM to establish an monitoring session.

- **F6**: VM response back.
- **F7**: VM will start replicating UC A's output stream to UC B.
- **F8:** UC B may send suspend command to VMM to suspend the current UA session.
- **P8.5:** VMM will check whether UC B has privilege to suspend the UA session.
- **F9:** VMM will ask VM to suspend the current UA session.
- **F10:** UC A will be notified about the suspension of his/her session on UC B request.

Note: As an alternate flow for this flow, in addition to getting the output stream a Watcher may explicitly request permission to send the interactions command too.

D.3 Reconnection

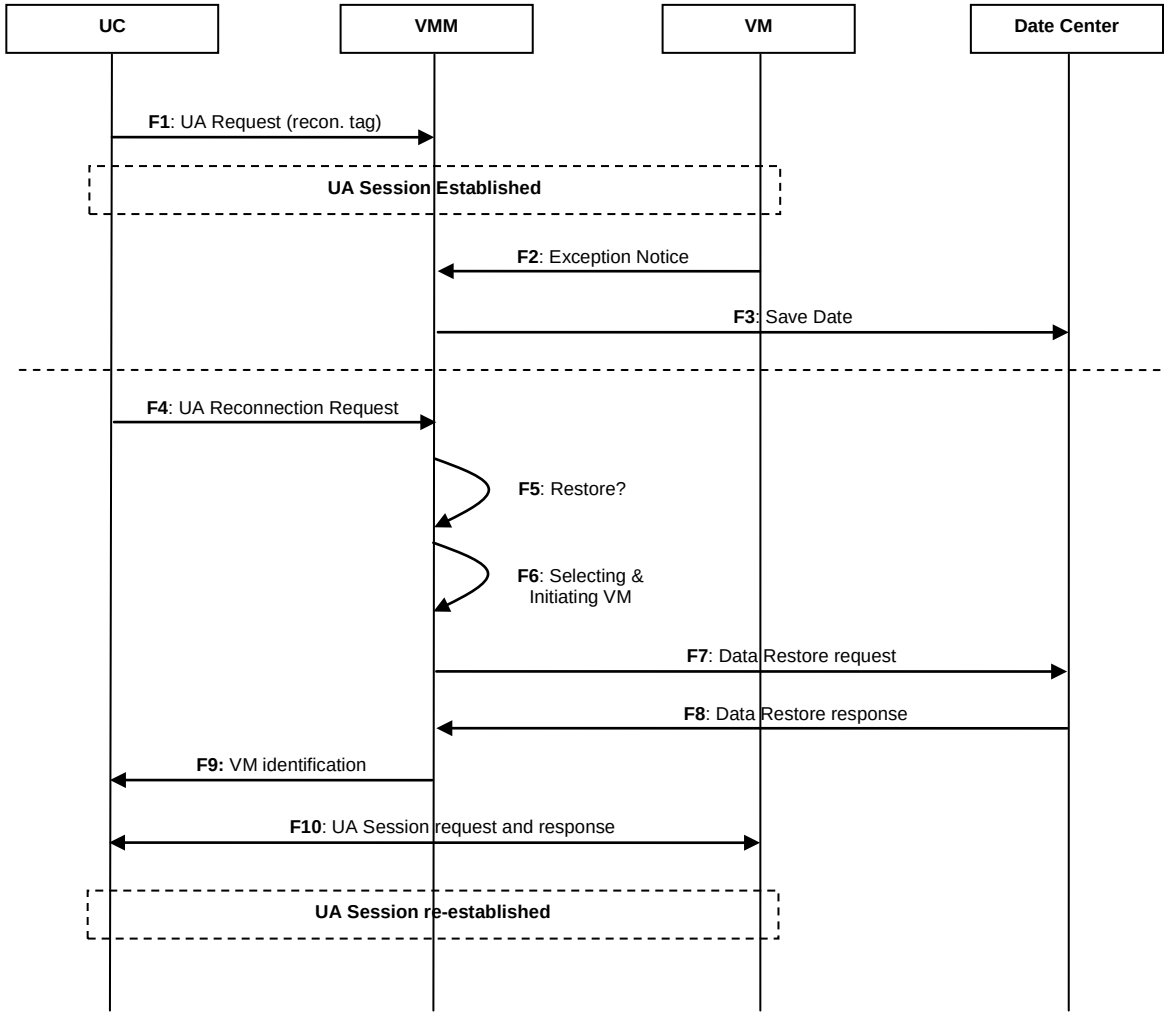

Figure 30

1. User sends the initial request to use particular UA, This request will contain an indication about the reconnection mechanism requested by the user to be used of this particular session.
2. At some point of time in future after user got disconnected VMM receives an exception notice from the VM, which was serving user, about the session disconnect. This notice include user id and UA id to identify the user and the application which was being used.

3. VMM save the VM Data as per the reconnection mechanism specified and then release the VM. The data will be saved as per the reconnection method i.e. in case of “normal reconnection” application data will be saved whereas for “no-loss reconnection” VM data will be saved.
4. Client tries to reconnect again with a normal UA request message.
5. Considering the user id and requested application, VMM will check for the previous records, if any.
6. VMM selects VM.
7. VMM restore the data from data centre.
8. Data gets restored.
9. VMM send selected VM identification to client.
10. Client connects with the notified VM to use the UA.

Appendix E. Application Download from TAS storefront

For downloading application from TAS storefront, VM shall implement TAS client [TAS] functionality for downloading an application. VM may use Application Download operation defined on TAS-2 interface in [TAS].

Appendix F. Media Capture mechanisms

F.1 Android

To capture the graphics of an application running on Android VM the following steps are taken:

- Open framebuffer Device. Normally, the device is "/dev/graphics/fb0"

```
if ((g_fbfd = open(pszDeviceName, O_RDONLY)) == -1)
{
  printf("init_fb: open framebuffer failed error.%d %s\n", errno,strerror(errno));
  return -1;
}
```

- Get frame information, including size and pixel, of framebuffer

```
//get framebuffer information
if (ioctl(g_fbfd, FBIOGET_VSCREENINFO, &fbScrinfo) != 0)
{
  printf("init_fb: ioctl(FBIOGET_VSCREENINFO) error.\n");
  close(g_fbfd);
  return -1;
}
// FB size
int pixels = fbScrinfo.xres_virtual * fbScrinfo.yres;
// frame size
int RGBFrameSize = pixels* 4;
int YUVFrameSize = (pixels*3)>>1;
// screen pixel, byte size
int bytespp = fbScrinfo.bits_per_pixel / 8;
```

- Map the framebuffer address with the user's memory address

```
//fb add -> user memory
unsigned char *g_fbmmap = (unsigned char *)mmap(NULL, RGBFrameSize*2, PROT_READ, MAP_SHARED, g_fbfd, 0);
```

- Calculate the address offset of framebuffer

```
// address offset
    f = (unsigned char*) (g_fbmmap + fbScrinfo.yoffset * fbScrinfo.xres_virtual * 4);
```

- Copy graphic pixels data from framebuffer

```
   // copy data
  memcpy(g_vncbuf, f, RGBFrameSize);
```

F.2 Windows

To capture the graphics of an application running on Android VM the following steps are taken:

- Get the window's handle of desktop: GetDesktop Window()
- Get window's Device Context: GetDC();
- Create DC and bitmap which compatible with the window's DC, and select the bitmap with the compatible DC:CreateCompatibleBitmap, CreateCompatibleDC(), SelectObject();
- Get graphic Data:
- Release object;