Virtualisation MO 1.0

Candidate Version 1.0 – 25 Jun 2013

Open Mobile Alliance

OMA-ER-VirMo-V1_0-20130625-C

Contents

Figures

Tables

1. Scope

This document describes the Virtualization MO that manages the Virtual Machines in the device by leveraging the OMA DM Protocol [DM13]. This document includes the Requirements, Architecture, and Technical Specification for the Virtualization MO 1.0.

2. References

2.1 Normative References

[DCMO] "Device Capability Management Object", Version 1.0, Open Mobile Alliance™, OMA-ERELD-DCMO-V1_0, URL:http://www.openmobilealliance.org

[DLOTA] "Generic Content Download Over The Air Specification", Version 1.0, Open Mobile Alliance™, OMA-Download-OTA-V1_0, URL:http://www.openmobilealliance.org

[DM13] "OMA Device Management", Version 1.3. Open Mobile Alliance™, OMA-ERELD-DM-V1_3-20120306-C, URL:http://www.openmobilealliance.org

[DMDICT] "OMA Device Management Dictionary", Version 1.0, Open Mobile Alliance™, OMA-SUP-DM_Dictionary-V1_0, URL:http://www.openmobilealliance.org

[FUMO] "Firmware Update Management Object", Version 1.0, Open Mobile Alliance™, OMA-ERELD-FUMO-V1_0, URL:http://www.openmobilealliance.org

[GwMO] "Gateway Management Object", Version 1.0, Open Mobile Alliance™, OMA-ERELD-GwMO-V1_0, URL:http://www.openmobilealliance.org

[LAWMO] "Lock and Wipe Management Object", Version 1.0, Open Mobile Alliance™, OMA-ERELD-LAWMO-V1_0, URL:http://www.openmobilealliance.org

[OSE] "OMA Service Environment", Open Mobile Alliance™, 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, 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

[SCOMO] "Software Component Management Object", Version 1.0, Open Mobile Alliance™, OMA-ERELD-SCOMO-V1_0, URL:http://www.openmobilealliance.org

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

2.2 Informative References

[OMADICT] "Dictionary for OMA Specifications", Version 2.9, Open Mobile Alliance™, OMA-ORG-Dictionary-V2_0, 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

Virtual Machine	A Virtual Machine is an isolated virtual execution environment on a device.
Virtualization Enabled Device	A device with the capabilities to manage the Virtual Machines in the device. This capability can be provisioned to the device at the factory or by firmware updates.
Virtual Machine Suspending	The operation of stopping the execution of the Virtual Machine. The current state of the Virtual Machine is saved. Once suspended, the allocated resources such as RAM, CPU used by the Virtual Machine are released
Virtual Machine Resuming	The operation of continuing the execution of the suspended Virtual Machine from the saved state. The necessary resources are reallocated for the Virtual Machine

Kindly consult **Error! Reference source not found.** for more definitions used in this document.

3.3 Abbreviations

Kindly consult **Error! Reference source not found.** for all definitions used in this document.

4. Introduction

This specification enables the management of any Virtual Machine on a device. A typical use case of Virtual Machines on a device is an enterprise Virtual Machine to provide a secure and separate workspace environment for employees. Basic management operations include creating/deleting, resuming/suspending and powering on/off of a Virtual Machine. In addition, advanced management features supported by this enabler include resource allocation to a Virtual Machine. This specification also supports to reuse Management Objects to provide the device managements for the Virtual Machine. For example, DCMO [DCMO] can be used for the peripheral access management of a Virtual Machine, and LAWMO [LAWMO] can be used for the locking & unlocking a Virtual Machine.

The Virtualization MO leverages hardware virtualization technologies on mobile devices, which virtualizes hardware resources such as CPU, RAM and storage so as to provide the abstract hardware layer. The hypervisor is one of such virtualization technologies, and can operate multiple Virtual Machines on top of it. Each Virtual Machine may run a guest operating system that does not necessarily know whether it runs on real hardware or on top of the Virtual Machine. A hypervisor can be one of two types; Type I and Type II. The Type I hypervisor runs directly on the device hardware, while the Type II hypervisor runs within an operating system. This specification can be used to manage devices running both types of hypervisors.

5. Requirements (Normative)

5.1 High-Level Functional Requirements

Label	Description	Release
VirMO-HLF-01	The enabler SHALL support a mechanism to create a new Virtual Machine to the device.	1.0
VirMO-HLF-02	The enabler SHALL support a mechanism to delete a Virtual Machine from the device.	1.0
VirMO-HLF-03	The enabler SHALL support a mechanism to power on a Virtual Machine.	1.0
VirMO-HLF-04	The enabler SHALL support a mechanism to power off a Virtual Machine.	1.0
VirMO-HLF-05	The enabler SHALL support a mechanism to manage device resources allocated to a Virtual Machine.	1.0
VirMO-HLF-06	The enabler SHALL support a mechanism to manage peripheral access to a Virtual Machine.	1.0
VirMO-HLF-07	The VirMO enabler SHALL collaborate with the existing MOs to provide the corresponding functionalities for each Virtual Machine.	1.0
VirMO-HLF-08	The VirMO enabler SHALL support a mechanism to provision firmware to a Virtual Machine.	1.0
VirMO-HLF-09	The VirMO enabler SHALL support a unique ID to identify a Virtual Machine within the device.	1.0
VirMO-HLF-10	The VirMO enabler SHALL support a mechanism to suspend a Virtual Machine.	1.0
VirMO-HLF-11	The VirMO enabler SHALL support a mechanism to resume a Virtual Machine.	1.0
VirMO-HLF-12	The VirMO enabler SHALL support a mechanism to lock a Virtual Machine.	1.0
VirMO-HLF-13	The VirMO enabler SHALL support a mechanism to unlock a Virtual Machine.	1.0
VirMO-HLF-14	The enabler SHALL support a mechanism to update the firmware of an existing Virtual Machine on the device.	1.0
VirMO-HLF-15	The VirMO enabler SHALL support a mechanism to deliver the Virtual Machine image to the device.	1.0

6. Architectural Model

6.1 Dependencies

The Virtualization MO architecture diagram indicates dependencies on the OMA DM [DM13] architecture.

6.2 Architectural Diagram

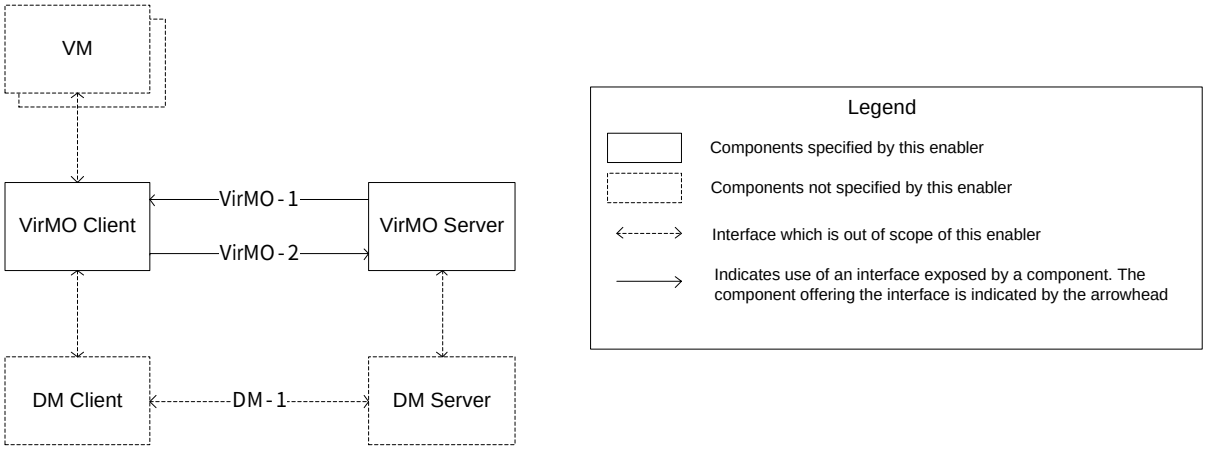

Figure 1: Virtualization MO Architecture Model

6.3 Functional Components and Interfaces/reference points definition

6.3.1 Components

6.3.1.1 VirMO Client

The VirMO Client is a logical entity, which is responsible for executing Virtual Machine management operations from the VirMO Server. The VirMO Client responds with the results of those operations, and issues Generic Alerts to the VirMO Server. The VirMO Client uses device internal interfaces to manage the Virtual Machines. The VirMO Client leverages the DM Client to communicate with the VirMO Server.

6.3.1.2 VirMO Server

The VirMO Server is a logical entity, which is dedicated to issue Virtual Machine management operations to the VirMO Client, and to consume the Generic Alerts from the VirMO Client. The VirMO Server leverages the functionalities provided by the DM Server to communicate with the VirMO Client.

6.3.1.3 Virtual Machine

The Virtual Machine is the isolated execution environment that can be managed by the VirMO Server. The VirMO Client manipulates the Virtual Machine through the device internal interface according to the instructions given from the VirMO Server.

6.3.1.4 DM Client

The OMA DM Client is defined in the [DM13] enabler and is the subject of those specifications.

6.3.1.5 DM Server

The OMA DM Server is defined in the [DM13] enabler and is the subject of those specifications.

6.3.2 Interfaces

6.3.2.1 VirMO-1 Interface

The VirMO-1 interface allows the VirMO Server to invoke the Virtual Machine management operations to the VirMO Client, and to receive the status and results from the VirMO Client. This interface uses the underlying DM-1 interface.

6.3.2.2 VirMO-2 Interface

The VirMO-2 interface allows the VirMO Client to send Generic Alerts to the VirMO Server using the underlying DM-1 interface.

6.3.2.3 DM-1 Interface

The DM-1 interface is defined in the [DM13] enabler and is the subject of those specifications. It provides a generic interface over which device management commands, status and Generic Alerts are exchanged between the VirMO Server and the VirMO Client.

6.4 Virtualization MO and Type I and II Hypervisor (Informative)

In this section, illustrative examples describe how Virtualization MO can be adapted into devices with different types of hypervisors (Type I and Type II). The examples are only for the understanding of the overall interactions between the hypervisor and Virtualization MO, and other approaches are not precluded.

6.4.1 Type I Hypervisor

The Type I hypervisor runs directly on the device hardware, and operates multiple Virtual Machines on top of it. One of the Virtual Machines is called as the Management VM that takes the role of managing other Virtual Machines. The Management VM may run a simple OS providing only limited functionalities just enough to fulfil the purposes. The VirMO Client and the DM Client are installed in this Management VM, and the VirMO Client interacts with the hypervisor and Virtual Machines to accomplish the operations from the VirMO Server.

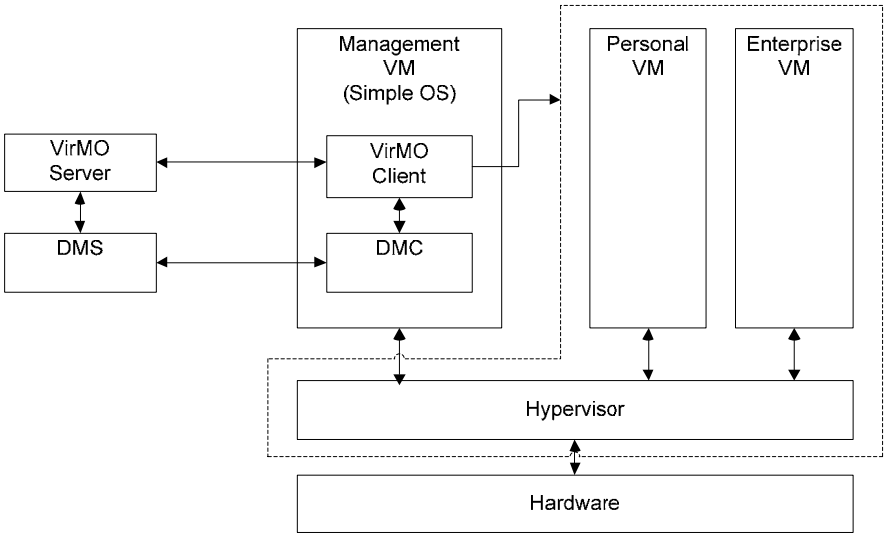

Figure 2: Virtualization MO with the Type I Hypervisor

6.4.2 Type II Hypervisor

The Type II hypervisor runs within a conventional operating system. The hypervisor also operates the multiple Virtual Machines on top of it just as the Type I hypervisor. The VirMO Client and the DM Client are installed within the conventional OS, and the VirMO Client interacts with the hypervisor and the Virtual Machines to accomplish the operations from the VirMO Server.

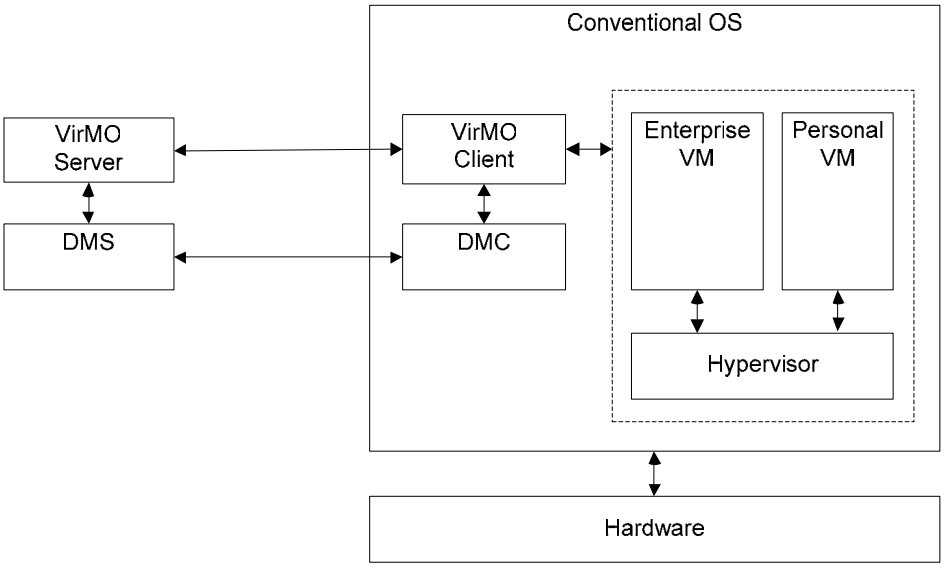

Figure 3: Virtualization MO with the Type II Hypervisor

7. Virtual Machine Management Framework

7.1 VM State Machine and Primitives

Using the management operations, the VirMO Server can manage the Virtual Machine in the device. The management operations may include creating, deleting and suspending the Virtual Machine, and the state of the Virtual Machine can be changed as the results of the management operations. The state of the Virtual Machine represents the present condition of the Virtual Machine, and the VM state machine shows the possible states and the primitives regarding the Virtual Machine.

The Virtual Machine has the well-defined states, and the state transition is triggered by the VM management primitives. The state transitions MUST be atomic; if a state transition fails, the VirMO Client MUST reverse the operation and MUST make sure the VM remains in the previous state. The following figure shows the state transition triggered by VM management primitives:

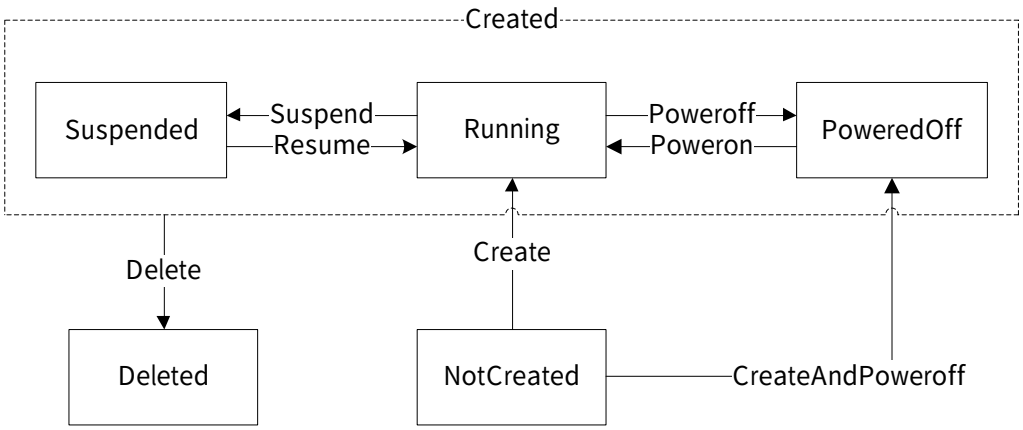

Figure 4: VM State Machine

7.1.1 NotCreated State

In NotCreated state, the VM has not been created in the device. Information used for creating the VM can be pre-configured in this state.

The VirMO Client MUST support this state.

7.1.2 Created State

The Running, Suspended and PoweredOff states are collectively grouped into the Created state. In Created state, the VM has been created, which means that a virtual execution environment for the VM is set up and ready for running.

7.1.2.1 Running State

In Running state, the VM has been created, which means that a virtual execution environment for the VM is set up and running. The VM allocates the required resources for the operating system and applications to be properly installed and executed.

The VirMO Client MUST support this state.

7.1.2.2 Suspended State

In Suspended state, the VM has been created, but the execution of the VM is stopped. All applications and the operating system in the VM MUST NOT be executed, and the current execution state of the VM must be saved. Once suspended, the

allocated resources for the VM may be released for other utilizations, but MUST be reallocated when the VM is transitioned to the Running state again.

The VirMO Client SHOULD support this state.

7.1.2.3 PoweredOff State

In PoweredOff state, the VM has been created, but the VM is powered off. When the VM is powered off, the operating system or applications running on the VM terminate together. To make the VM running again, powering on the VM through the normal operating system boot process is required.

The VirMO Client MUST support this state.

7.1.3 Deleted State

This is a logical state. In Deleted state, the VM has been deleted from the device, and the resources allocated to the VM are released and inaccessible to any entities. Data that belong to the VM MAY be destroyed.

The VirMO Client MUST support this state.

7.1.4 VM Management Primitives

VM management primitives are used for triggering the state transition of the VM. The primitive MUST be atomic. The explanations for each primitive are detailed as follows:

Primitives	Descriptions	Applicable State	Post-Primitive State	VirMO Client Support
Create	To create the VM. The created VM is in the Running state	NotCreated	Running	MUST support
CreateAndPowerOff	To create the VM. The created VM is in the PoweredOff state.	NotCreated	PoweredOff	MUST support
Suspend	To suspend the VM	Running	Suspended	MUST support if the Suspended state is supported
Resume	To resume the VM making the VM running	Suspended	Running	MUST support if the Suspended state is supported
Poweron	To power-on the VM	PoweredOff	Running	MUST support
Poweroff	To power-off the VM	Running	PoweredOff	MUST support
Delete	To delete the VM from the device	Created	Deleted	MUST support

8. Virtualization Management Object

The Virtualization MO provides the interface between the VirMO Server and the VirMO Client, which can be used for managing Virtual Machines. This section provides the pictorial description of the Virtualization MO and the explanations for each parameter.

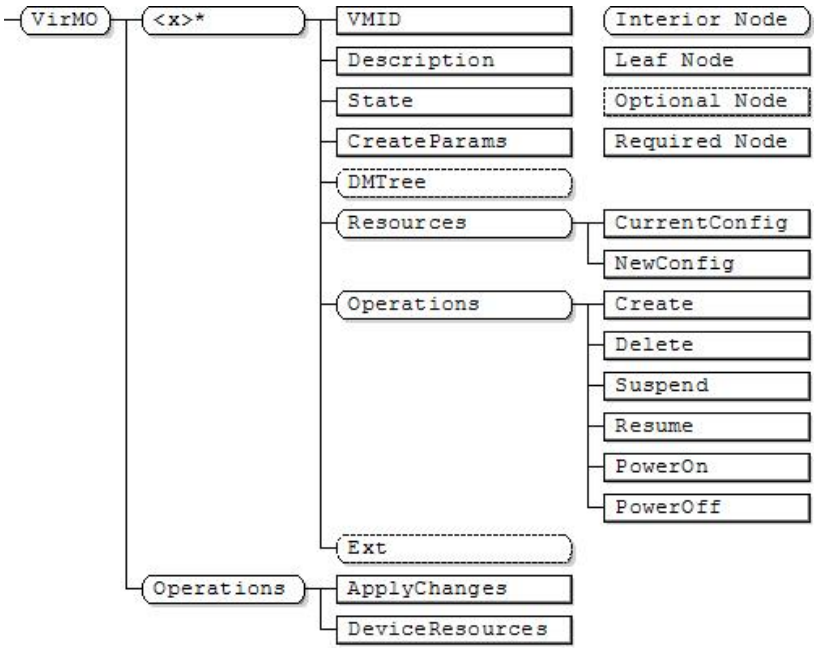

Figure 5: The Virtualization Management Object

VirMO

Status	Tree Occurrence	Format	Min. Access Types
Required	One	node	Get

This interior node is the root node for VirMO. The ancestor elements of this node define the position in the DM Tree for Virtualization MO.

The MOID for the Virtualization MO MUST be: “urn:oma:mo:oma-virmo:1.0”.

VirMO/<x>

Status	Tree Occurrence	Format	Min. Access Types
Required	ZeroOrMore	node	Get

This interior node is the placeholder for parameters regarding a particular Virtual Machine. For each Virtual Machine to be managed, corresponding parameters will be available under this interior node.

VirMO/<x>/VMID

Status	Tree Occurrence	Format	Min. Access Types
Required	One	chr	Get, No Replace

This leaf node specifies the identifier for the Virtual Machine. The value for this leaf node MUST be assigned by the VirMO Client and it MUST be unique within the Virtualization MO.

VirMO/<x>/Description

Status	Tree Occurrence	Format	Min. Access Types
Required	One	chr	Get

This leaf node specifies the description for the Virtual Machine.

VirMO/<x>/State

Status	Tree Occurrence	Format	Min. Access Types
Required	One	int	Get, No Replace

This leaf node specifies the current state of the Virtual Machine. This value MUST be set only by the VirMO Client, and MUST have one of the following values:

Valid Value	Descriptions
0	NotCreated state
1	PoweredOff state
2	Running state
3	Suspended state
4	Deleted state

VirMO/<x>/CreateParams

Status	Tree Occurrence	Format	Min. Access Types
Required	One	chr	Get

This node specifies the parameters used for creating the VM. The parameters MUST be organized as the sequence of “key=value” pairs, and the pairs MUS be separated by an ampersand “&”. The standard parameters are specified in the below table, but the vender-specific extensions for parameters are also possible. The VirMO Server and the VirMO Client MUST support all the parameters specified in the table.

Parameters	Descriptions	Occurrence
ImgID=<img_id>	This parameter specifies the VM image used for creating the VM. The <img_id> value MUST be one of the values from the *VMImgMO/<x>/ImgID* nodes.	One
Running=<bool>	If <bool> is true, the created VM MUST be in the Running state. If <bool> is false, the created VM MUST be in the PoweredOff state. If not specified, the default value MUST be true.	ZeroOrOne

VirMO/<x>/DMTree

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	node	Get

This interior node is the placeholder for the DM Tree of this Virtual Machine. The necessary Management Objects to provide the device management operations for the Virtual Machine MUST be instantiated under this node. Please refer to the section 11.2 for details.

This optional interior node is only valid when the device management with the shared DM Client is used.

VirMO/<x>/Resources

Status	Tree Occurrence	Format	Min. Access Types
Required	One	node	Get

This interior node groups the nodes for managing the resource allocation to this Virtual Machine.

VirMO/<x>/Resources/CurrentConfig

Status	Tree Occurrence	Format	Min. Access Types
Required	One	chr	Get, No Replace

This node specifies the current configuration of device resources that are allocated to this specific Virtual Machine. The parameters MUST be organized as a sequence of “key=value” pairs, and the pairs MUST be separated by an ampersand character “&”. The standard parameters are specified in the below table, but additional vender-specific parameters are allowed. The VirMO Server and the VirMO Client MUST support all the parameters specified in the table.

Parameters	Descriptions	Occurrence
RAM=<int>	This parameter specifies the amount of RAM in bytes that are allocated to this Virtual Machine.	One
Storage=<int>	This parameter specifies the amount of data partition flash storage in bytes that are allocated to this Virtual Machine.	One
CPU=<int>	This parameter specifies the device CPU time slice in percentage (0-100) that is dedicated to this Virtual Machine.	One
Network=<int>	This parameter specifies the bandwidth allocated to this Virtual Machine, expressed in percentage (0-100) of the bandwidth available to the device. The value MUST apply to both upstream and downstream bandwidth. The value MUST apply to all connection types.	One

VirMO/<x>/Resources/NewConfig

Status	Tree Occurrence	Format	Min. Access Types
Required	ZeroOrOne	chr	Add

This node specifies the configuration of device resources that MUST be allocated to this specific Virtual Machine when the VirMO Server sends an Exec command to the *ApplyConfig* node. The format defined in the *CurrentConfig* node MUST be applied to this node as well.

VirMO/<x>/Operations

Status	Tree Occurrence	Format	Min. Access Types
Required	One	node	Get

This interior node is a parent node for operations supported by the Virtualization MO.

VirMO/<x>/Operations/Create

Status	Tree Occurrence	Format	Min. Access Types
Required	One	null	Exec

This node is used with Exec command for the VirMO Server to create the Virtual Machine. Prior to the Exec command, the parameters for creating the VM MUST be provisioned at the *VirMO/<x>/CreateParams* node, and the VirMO Client MUST create the VM according to the parameters.

VirMO/<x>/Operations/Delete

Status	Tree Occurrence	Format	Min. Access Types
Required	One	null	Exec

This node is used with Exec command for the VirMO Server to delete the Virtual Machine. Only Virtual Machines in the Created state can be deleted.

VirMO/<x>/Operations/Suspend

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	null	Exec

This node is used with Exec command for the VirMO Server to suspend the Virtual Machine.

VirMO/<x>/Operations/Resume

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	null	Exec

This node is used with Exec command for the VirMO Server to resume the Virtual Machine.

VirMO/<x>/Operations/PowerOn

Status	Tree Occurrence	Format	Min. Access Types
Required	One	null	Exec

This node is used with Exec command for the VirMO Server to power on the Virtual Machine.

VirMO/<x>/Operations/PowerOff

Status	Tree Occurrence	Format	Min. Access Types
Required	One	null	Exec

This node is used with Exec command for the VirMO Server to power off the Virtual Machine.

VirMO/<x>/Ext

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	node	Get

This interior node is for vendor-specific extensions to store the Virtual Machine related information.

VirMO/Operations

Status	Tree Occurrence	Format	Min. Access Types
Required	One	node	Get

This interior node is a parent node for operations supported by the Virtualization MO. Unlike operations under the *VirMO/<x>/Operations* node, the operations under this interior node do not belong to a specific Virtual Machine.

VirMO/Operations/ApplyChanges

Status	Tree Occurrence	Format	Min. Access Types
Required	One	null	Exec

This node is used with Exec command for the VirMO Server to apply the resource changes made to one or several Virtual Machines as specified in the *VirMO/<x>/Resources/NewConfig* node. Virtual Machines that are associated with empty *NewConfig* nodes MUST NOT have resource configuration changes applied to them. After the changes have been applied successfully, the NewConfig nodes for all Virtual Machines MUST be deleted by the VirMO Client. The VirMO Client MUST check the consistency (for example, make sure that the sum of the CPU allocations for all Virtual Machines is 100%) of the new resource configurations before applying the changes.

VirMO/DeviceResources

Status	Tree Occurrence	Format	Min. Access Types
Required	One	chr	Get

This node specifies the total amount of device resources. The parameters MUST be organized as a sequence of “key=value” pairs, and the pairs MUST be separated by an ampersand character “&”. The standard parameters are specified in the below table, but additional vender-specific parameters are allowed. The VirMO Server and the VirMO Client MUST support all the parameters specified in the table.

Parameters	Descriptions	Occurrence
RAM=<int>	This parameter specifies the amount of RAM in bytes for the device.	One
Storage=<int>	This parameter specifies the amount of flash storage in bytes for the device.	One

9. Virtual Machine Image Management Object

Virtual Machine Image MO (VM Image MO) provides the interface between the VirMO Server and the VirMO Client, which can be used for managing Virtual Machine images. The VM image serves as an install image to create the Virtual Machine. This section provides the pictorial description of the VM Image MO and the explanations for each parameter.

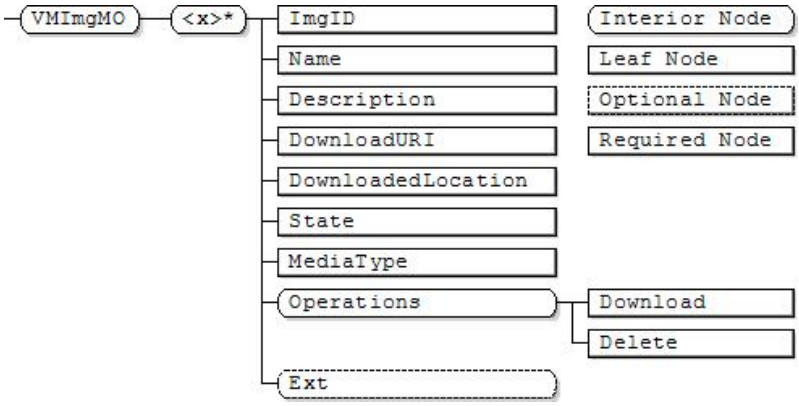

Figure 6: The VM Image Management Object

VMImgMO

Status	Tree Occurrence	Format	Min. Access Types
Required	One	node	Get

This interior node is the root node for the VM Image MO. The ancestor elements of this node define the position in the DM Tree for this MO.

The MOID for the VM Image MO MUST be: “urn:oma:mo:oma-vmimgmo:1.0”.

VMImgMO/<x>

Status	Tree Occurrence	Format	Min. Access Types
Required	ZeroOrMore	node	Get

This interior node is the placeholder for parameters regarding a particular VM image. For each VM image to be managed, corresponding parameters will be available under this interior node.

VMImgMO/<x>/ImgID

Status	Tree Occurrence	Format	Min. Access Types
Required	One	chr	Get

This leaf node specifies the identifier for the VM image. The value for this leaf node MUST be assigned by the VirMO Client and MUST be unique within this MO.

VMImgMO/<x>/Name

Status	Tree Occurrence	Format	Min. Access Types
Required	One	chr	Get

This leaf node specifies the name for the VM image (e.g., android_2.2_froyo). The value of this node MUST be set by the VirMO Server.

VMImgMO/<x>/Description

Status	Tree Occurrence	Format	Min. Access Types
Required	ZeroOrOne	chr	Get

This leaf node specifies human readable descriptions for the VM image.

VMImgMO/<x>/DownloadURI

Status	Tree Occurrence	Format	Min. Access Types
Required	One	chr	Get

This leaf node specifies the URI that can be used for downloading the VM image. The URI in this node is used for alternative download mechanisms (such as HTTP GET [RFC2616] or Download Over the Air [DLOTA]).

VMImgMO/<x>/DownloadedLocation

Status	Tree Occurrence	Format	Min. Access Types
Required	ZeroOrOne	chr	Get, No Replace

The value of this node specifies the local path for the completely downloaded VM image (e.g., /sdcard/download/virmo/images/android_2.2_froyo). The value of this node is valid only if the value of the *VMImgMO/<x>/State* node is 2 (Downloaded). The format of this value is platform specific, and out-of-scope of this specification. The value of this node MUST be set only by the VirMO Client.

VMImgMO/<x>/State

Status	Tree Occurrence	Format	Min. Access Types
Required	One	int	Get, No Replace

This leaf node specifies the current state of the VM image. This value MUST be set only by the VirMO Client, and MUST have one of the following values:

Integer Value	Descriptions
0 (Not Downloaded)	The download is not started, and no data is received.
1 (Downloading)	The download has been started.
2 (Downloaded)	The download is successfully finished, and all data has been received.
3 (Download Failed)	The download is failed, and 0 or more bytes of data might be received.

VMImgMO/<x>/MediaType

Status	Tree Occurrence	Format	Min. Access Types
Required	One	chr	Get

This leaf node specifies the media type of the VM image. The value MUST be a MIME media type. The value of this node MUST be set by the VirMO Server.

VMImgMO/<x>/Operations

Status	Tree Occurrence	Format	Min. Access Types
Required	One	node	Get

This interior node is the parent node for operations supported by this MO.

VMImgMO/<x>/Operations/Download

Status	Tree Occurrence	Format	Min. Access Types
Required	One	null	Get

This node is used with Exec command to download the VM image using the *VMImgMO/<x>/DownloadURI* node. After completely downloading the VM image, the VirMO Client MUST properly set the *VMImgMO/<x>/DownloadedLocation* node, and MUST send the *VM Image Ready Alert* back to the VirMO Server to inform the completion of the download.

VMImgMO/<x>/Operations/Delete

Status	Tree Occurrence	Format	Min. Access Types
Required	One	null	Get

This node is used with Exec command to delete the VM image. In case that the VM image is completely or partially downloaded, the downloaded image MUST be deleted. The corresponding sub-tree under the *VMImgMO/<x>* node MUST be deleted.

VMImgMO/<x>/Ext

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	node	Get

This interior node is for vendor-specific extensions to store the VM image related information.

10.VM Management Operation Procedures

This section describes the procedures for the Virtual Machine management operations.

10.1 Delivering the VM Image

The VirMO Server delivers the VM image in the device as follows:

Step 1: The VirMO Server sends the request to the VirMO Client for creating the sub-tree under the *VMImgMO* node.

Step 2: After receiving the request from the VirMO Server, the VirMO Client creates the sub-tree under the *VMImgMO* node. The VirMO Client MUST set the value of the *VMImgMO/<x>/ImgID* and MUST set the value 0 (Not Downloaded) to the *VMImgMO/<x>/State* node. After successfully processing the request, the VirMO Client sends the results (i.e., status code) to the VirMO Server.

Step 3: The VirMO Server MUST properly set parameters used for delivering the VM image. Then, the VirMO Server sends the Exec command to the *VMImgMO/<x>/Operations/Download* node.

Step 4: After receiving the Exec command, the VirMO Client starts to download the VM image from the URI stored at the *VMImgMO/<x>/DownloadURI* node. Alternative download mechanisms such as HTTP [RFC2616] or Download Over the Air [DLOTA] are used to download the VM image. The VirMO Client MUST properly update the *VMImgMO/<x>/State* node. When the VM image is completely downloaded, the VirMO Client MUST set the *VMImgMO/<x>/DownloadedLocation* node, and MUST send the *Image Ready Alert* to the VirMO Server. The VirMO Client responds to the VirMO Server by using either synchronous or asynchronous reporting mechanism.

10.2 Creating Virtual Machine

The VirMO Server can create the Virtual Machine in the device as follows:

Step 1: The VirMO Server sends the request to the VirMO Client for creating the sub-tree under the *VirMO* node.

Step 2: After receiving the request from the VirMO Server, the VirMO Client creates the sub-tree under the *VirMO* node. The VirMO Client MUST assign a value to the *VirMO/<x>/VMID* node and MUST set the value 0 (NotCreated state) to the *VirMO/<x>/State* node. After successfully processing the request, the VirMO Client sends the results (i.e., status code) to the VirMO Server.

Step 3: The VirMO Server MUST properly set the *VirMO/<x>/CreateParams* node. Then, the VirMO Server sends Exec command against the *VirMO/<x>/Operations/Create* node to create a Virtual Machine.

Step 4: After receiving the Exec command, the VirMO Client creates a new Virtual Machine according to the *VirMO/<x>/CreateParams* node, and MUST update the *VirMO/<x>/State* node. Then the VirMO Client can respond to the VirMO Server by using either synchronous or asynchronous reporting mechanism.

Step 5: After the Virtual Machine is successfully created by the VirMO Client, the VirMO Server MAY delete the sub-tree under the *VMImgMO* node.

10.3 Deleting Virtual Machine

The VirMO Server can delete the Virtual Machine in the device as follows:

Step 1: The VirMO Server sends the Exec command to the *VirMO/<x>/Operations/Delete* node.

Step 2: After receiving the Exec command, the VirMO Client MUST delete the Virtual Machine in the device and the corresponding sub tree under the *VirMO/<x>* node.

11.Device Management for Virtual Machines

A Virtual Machine created in the device can be regarded as another independent device that an operating system and applications are running in. A Virtual Machine as a separate device can be managed by a remote server for firmware update, software management, diagnostics, remote control, configurations, etc. This kind of management is called device management of the Virtual Machine, and is very similar to normal device management enabled by the DM Protocol [DM1.3] and Management Objects.

Device management of a Virtual Machine can be also leveraged by Management Objects (e.g., FUMO [FUMO], SCOMO [SCOMO], LAWMO [LAWMO], DCMO [DCMO], etc.). For example, DCMO can be used to control the peripherals of the Virtual Machine; therefore the USB and Camera capability for the Virtual Machine can be enabled and disabled by the VirMO Server. To utilize Management Objects for the device management of a Virtual Machine, the DM Client may or may not be running in the Virtual Machine. In this section, both approaches to utilize Management Objects are explained.

11.1 Device Management with Separate DM Client

To provide device management for the Virtual Machine, this approach requires that a separate DM Client runs in the Virtual Machine. The DM Server can provide the device management for the Virtual Machine by directly communicating with the DM Client in the VM or by indirectly communicating with the DM Client in the VM by using the Gateway Management Object [GwMO].

11.1.1 Direct Device Management

In Figure 7, Personal VM and Enterprise VM are created by the VirMO Server and the VirMO Server needs to provide the device management for the Enterprise VM. For the direct device management, the DM Client needs to be installed in the Enterprise VM, and the DM Server in the right side can directly communicate with the DM Client in the Enterprise VM. The Enterprise VM is seen as a separate device from the DM Server perspective.

The DM Client is not installed in the Personal VM since it is not required to provide the device management for the Personal VM. In case that the VirMO Server needs to provide the direct device management for the Personal VM, then the DM Client needs to be installed, and the Personal VM can be seen as another separate device from the DM Server perspective.

Figure 7 gives the illustrative example for the direct device management based on the Type I Virtualization, but it is also possible to apply this approach to the Type II Virtualization.

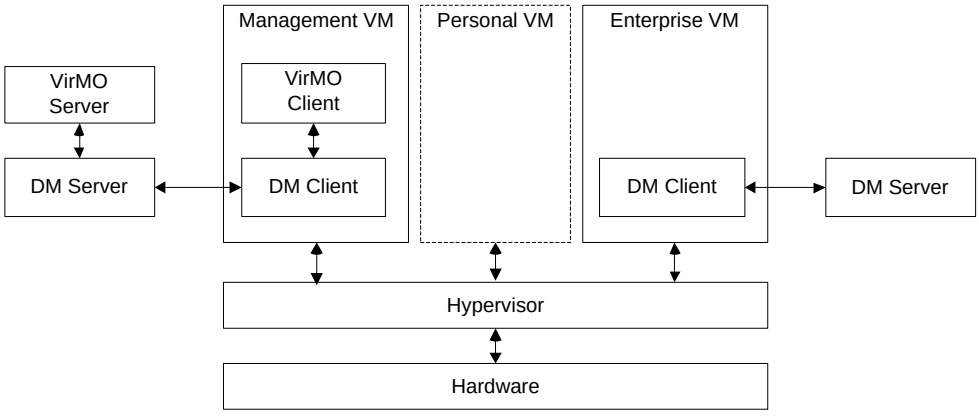

Figure 7: Illustrating Direct Device Management with Type I Virtualization

11.1.2 Indirect Device Management using GwMO

In Figure 8, the VirMO Server needs to provide device management for the Enterprise VM, but the DM Client in the Enterprise VM cannot directly communicate with the remote DM Server. In this case, the VirMO Server can provide device management for the Enterprise VM by using the GwMO [GwMO].

Both Transparent and Proxy Mode can be used to provide the indirect device management for the VM. In case of the Transparent Mode, the DM Notification is delivered to the DM Client in the VM by the aid of the GwMO Component in the device. After receiving the DM Notification, the DM Client in the VM initiates the DM session with the DM Server, and the device management for the VM is performed between the DM Server and the DM Client.

For the Proxy Mode, two related DM sessions are established; one is between the DM Server and the DM Gateway (working as the DM Client), and the other is between the DM Gateway (working as the DM Server) and the Virtual Machine. Since the DM Client in the VM does not need to have a direct communication with the DM Server, the DM Client can have the device identifier that is locally unique within the device.

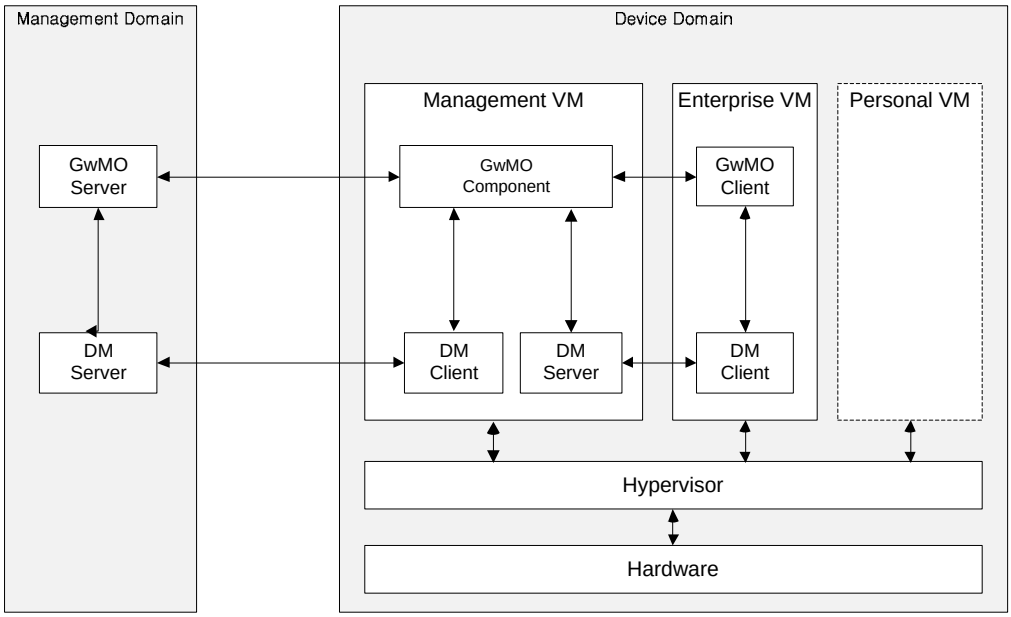

Figure 8: Illustrating Indirect Device Management with Type I Virtualization

11.1.3 Bootstrapping Virtual Machine

For the separated DM Client approach, the Virtual Machine needs to be bootstrapped for the proper operations. The DM Bootstrap specification [DM13] lists four possible approaches for the DM Bootstrap process: customized bootstrap, server initiated bootstrap, bootstrap from SmartCard, and client initiated bootstrap. Those DM bootstrapping approaches can be applied for the Virtual Machine as well. The bootstrap information might be prepared during the creation of the VM image and included in the VM image, which can be categorized as the customized bootstrap.

11.2 Device Management with Shared DM Client

Unlike the separated DM Client approach, this approach does not require a separate DM Client in the Virtual Machine, and the DM Client located in the Management VM can be used to provide the device management for other Virtual Machines. This approach can be applied to both Type I and Type II Virtualization, and is illustrated in Figure 9 using Type I Virtualization.

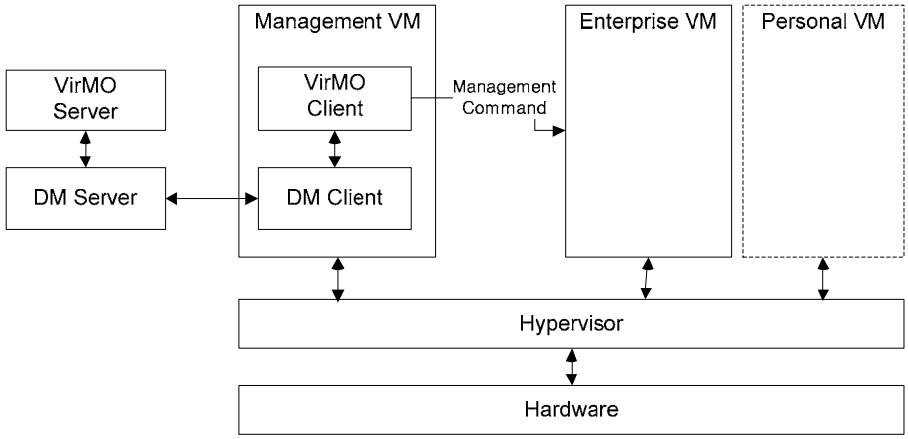

Figure 9: Illustrating the Approach based on the Type I Virtualization

In Figure 9, When the VirMO Server needs to provide the device management for the Enterprise VM, the DM Client in the Management VM can be used, and the VirMO Client forwards the device management command from the VirMO Server to other Virtual Machines. From the remote server perspective, there is only one device (i.e., the Management VM).

To forward the device management command to other Virtual Machines, the VirMO Client MUST keep the DM Tree for other Virtual Machines under the *VirMO/<x>/DMTree* interior node. The *VirMO/<x>/DMTree* works like a placeholder for the DM Tree of a particular Virtual Machine.

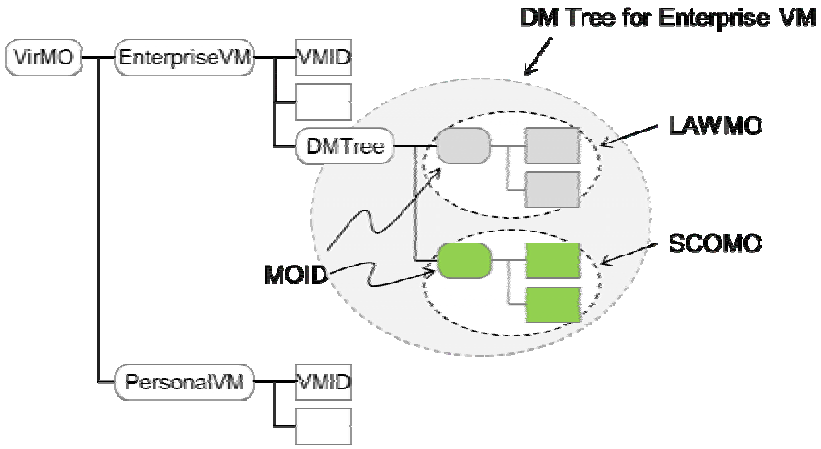

Figure 10: DM Tree for the Enterprise VM allocated under the *VirMO/<x>/DMTree* node

In Figure 10, the DM Server intends to provide device management for the Enterprise VM, and the DM Tree for the Enterprise VM is instantiated under the *VirMO/EnterpriseVM/DMTree* node. All the necessary Management Objects to provide the device management for the Virtual Machine MUST be instantiated under the *VirMO/<x>/DMTree* node, and in this example, the LAWMO and SCOMO sub-trees are created. The *VirMO/PersonalVM/DMTree* node for the Personal VM is not present since the DM Server does not intend to provide device management for the Personal VM. The DM Server can create the necessary Management Object sub-trees under the *VirMO/PersonalVM/DMTree* node in case device management for the Personal VM is required.

The VirMO Server can deliver the management command to the Virtual Machine using the sub-tree under the *VirMO/<x>/DMTree* node. For example, the Exec command targeting the *VirMO/EnterpriseVM/DMTree/FUMO/Update* node can be interpreted as the firmware update command for the Enterprise VM, and the Exec command for the *VirMO/EnterpriseVM/DMTree/LAWMO/Operations/FullyLock* node can be interpreted to fully lock the Enterprise VM.

11.2.1 Sending and Processing Device Management Command for VM

In this approach, the VirMO Server and the VirMO Client MUST follow below procedures to send and process the device management command for the Virtual Machine.

- **Step 1 (MO configuration under the *DMTree* node)**: After creating the Virtual Machine, the VirMO Server MUST configure Management Objects under the *VirMO/<x>/DMTree* node, which are necessary to provide device management for the Virtual Machine. The *VirMO/<x>/DMTree* node works as the root node for the DM Tree that belongs to a particular Virtual Machine. The MO root node located under the *VirMO/<x>/DMTree* node has the MOID that can be used to distinguish the type of each Management Object.
- **Step 2 (Sending device management commands targeting a particular Virtual Machine)**: Once Management Objects are configured under the *VirMO/<x>/DMTree* node, the VirMO Server can deliver the management commands using the configured Management Objects. For example, in case SCOMO is configured for the Enterprise VM, the Get command targeting the *VirMO/EnterpriseVM/DMTree/SCOMO/Inventory/Deployed* node can be sent to retrieve the installed software components for the Enterprise VM.
- **Step 3 (Processing the device management command for a particular Virtual Machine)**: When receiving the DM command, the VirMO Client can check whether the DM command is a device management command for a particular Virtual Machine. If the DM command targets a node under the *VirMO/<x>/DMTree*, then the VirMO Client MUST consider that the DM command is a device management command for a Virtual Machine identified by the *VirMO/<x>/VMID* node.

  For example, if the DM command targets the *VirMO/EnterpriseVM/DMTree/SCOMO/Inventory/Deployed* node, the VirMO Client finds out this DM command is for retrieving the installed software component of the Enterprise VM identified by the *VirMO/EnterpriseVM/VMID* node.

  After processing the device management command for a particular Virtual Machine, the VirMO Client can respond by using either synchronous or asynchronous reporting mechanism.

11.2.2 User Interaction for Virtual Machine

For the shared DM Client approach, user interaction commands specified in [DM13] are not applicable. In DM Protocol, user interaction command is sent from the DM Server to the DM Client as an Alert, and <Alert>/<Data> element specifies the Alert Type for a specific user interaction (i.e., Display, Confirmation, User Input, User Choice and Progress Notification). When receiving this user interaction Alert, the VirMO Client cannot distinguish that which Virtual Machine this user interaction targets. For the user interaction with the Virtual Machine, additional nodes for user interaction need to be added into the Virtualization Management Object.

Figure 11 shows the pictorial description of those nodes, and the detailed explanations for each parameter are follows.

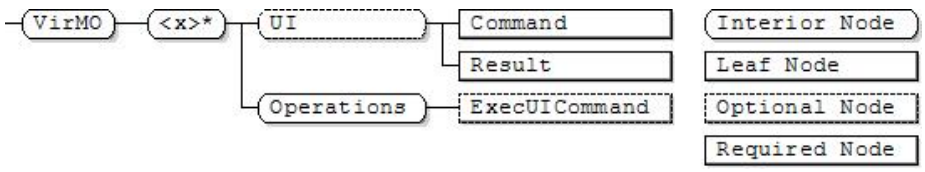

Figure 11: Nodes for Virtual Machine User Interaction

VirMO/<x>/UI

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	node	Get

This interior node is the parent node for the parameters pertaining to the user interaction for a particular Virtual Machine.

VirMO/<x>/UI/Command

Status	Tree Occurrence	Format	Min. Access Types
Required	One	xml	Get

This node contains user interaction command that targets a particular Virtual Machine identified by the *VirMO/<x>/VMID* nodes. The value of this node MUST conform to the structure of the <Alert> element for the user interaction command, as per the DM representation protocol [DM1.3].

VirMO/<x>/UI/Result

Status	Tree Occurrence	Format	Min. Access Types
Required	One	Xml	Get, No Replace

This node contains the result for the user interaction command stored in the *VirMO/<x>/UI/Command* node. The value of this node MUST conform to the structure of the <Status> element, as per the DM representation protocol [DM1.3].

VirMO/<x>/Operations/ExecUICmd

Status	Tree Occurrence	Format	Min. Access Types
Optional	One	Null	Exec

This node is used with Exec command to start the processing of the user interaction command stored in the *VirMO/<x>/UI/Command* node. Before executing this node, the VirMO Server MUST properly set the *VirMO/<x>/UI/Command* node. After executing the user interaction command in the *VirMO/<x>/UI/Command* node, the VirMO Client MUST set the *VirMO/<x>/UI/Result* node.

The following figure shows that the VirMO Server executes the user interaction command for the Virtual Machine using the *VirMO/<x>/UI* sub-tree.

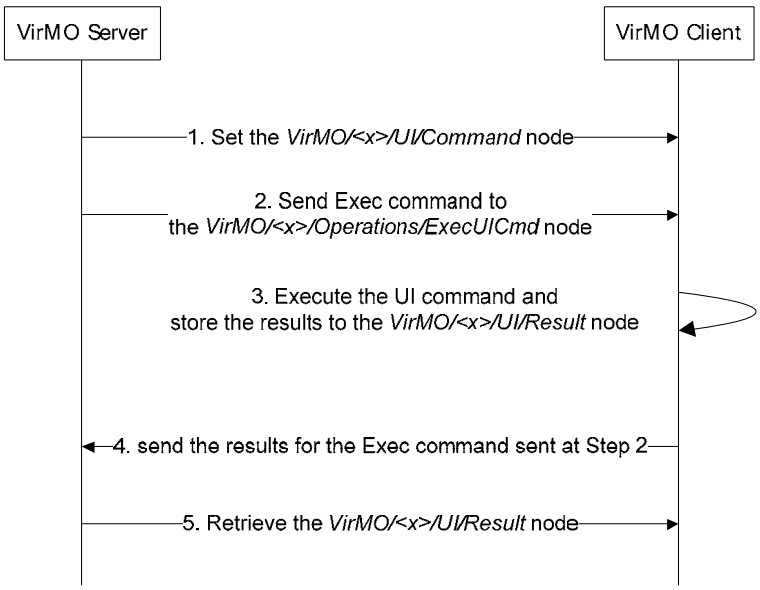

Figure 12: User Interaction with the Virtual Machine

- **Step 1**: The VirMO Server MUST set the *VirMO/<x>/UI/Command* node with the user interaction command. For example, if the VirMO Server wants to "Confirm or Reject" user interaction with the Enterprise VM, the value of the *VirMO/EnterpriseVM/UI/Command* node can be set by the VirMO Server as follows:

```
<Alert>
  <CmdID>2</CmdID>
  <Data>1101</Data>
  <Item></Item> <!-- no optional parameters -->
  <Item>
    <Data>Do you want to add the CNN access point?</Data>
  </Item>
</Alert>
```

- **Step 2**: The VirMO Server sends an Exec command to the *VirMO/<x>/Operations/ExecUICmd* node.

- **Step 3**: When receiving the Exec command to the VirMO/<x>/Operations/ExecUICmd node, the VirMO Client MUST execute the user interaction command in the *VirMO/<x>/UI/Command* node for the Virtual Machine identified by the *VirMO/<x>/VMID* node. After completing the user interaction command, the VirMO Client MUST set the *VirMO/<x>/UI/Result* node. For example, the value of the *VirMO/<x>/UI/Result* node for the above user interaction command can be as follows:

```
<Status>
  <CmdID>2</CmdID>
  <MsgRef>1</MsgRef>
  <CmdRef>2</CmdRef>
  <Cmd>Alert</Cmd>
  <Data>304</Data> <!-- Answer was “no” -->
</Status>
```

- **Step 4**: The VirMO Client sends the results for the Exec command that is sent at Step 2 (Exec to the *VirMO/<x>/Operations/ExecUICmd* node). The VirMO Client can respond by using either synchronous or asynchronous reporting mechanism.

- **Step 5**: On receiving the results for the Exec command, the VirMO Server retrieves the *VirMO/<x>/UI/Result* node to get the result for the user interaction command configured at the Step 1.

12. Alerts

This section describes the various alerts that have been defined in this specification.

12.1 Image Ready Alert

The *Image Ready Alert* is issued by the VirMO Client to the VirMO Server when the VirMO Client completely downloads the VM image. The *Image Ready Alert* conforms to the Generic Alert [DM1.3] mechanism, and the alert message includes the following data:

- <Meta>/<Type> element: Contains the media type of the alert content. The value MUST be the alert type identifier "urn:oma:at:oma-virmo:imageready:1.0".
- <Meta>/<Format> element: Contains the format of the alert content. The value MUST be "xml".
- <Meta>/<Mark> element: Contains the importance level of the alert message. This element is OPTIONAL.
- <Source>/<LocURI> element: Contains the source address of the MO. The value MUST be the address of the *VMImgMO/<x>/Operations/Download* node.

The following is an example of the *Image Ready Alert* which reports the completion of the VM image download:

```
<Alert>
     <CmdID>2</CmdID>
     <Data>1226</Data>  <!-- Generic Alert -->
     <Item>
          <Source><LocURI>./VMImgMO/entvm/Operations/Download</LocURI></Source>
          <Meta>
               <Type xmlns="syncml:metinf">
                    urn:oma:at:oma-virmo:imageready:1.0
               </Type>
               <Format xmlns="syncml:metinf">xml</Format>
               <Mark xmlns="syncml:metinf">informational</Mark>
          </Meta>
     </Item>
</Alert>
```

13.Release Information

13.1 Supporting File Document Listing

Doc Ref	Permanent Document Reference	Description
Supporting Files		
[VirMO_DDF]	OMA-SUP-MO-VirMO-V1_0	Virtualization MO Device Description File Working file in DM_MO directory: file: VirMO-V1_0.ddf path: http://www.openmobilealliance.org/tech/dtd/
[VMImgMO_DDF]	OMA-SUP-MO-VMImgMO-V1_0	Virtual Machine Image MO Device Description File Working file in DM_MO directory: file: VMImgMO-V1_0.ddf path: http://www.openmobilealliance.org/tech/dtd/

Table 1: Listing of Supporting Documents in Virtualization MO Release

13.2 OMNA Considerations

The OMNA registry needs to add and maintain the following in the MO registry:

MO Identifier	Description	Owner	Version	MO DDF	MO Spec
urn:oma:mo:oma-virmo:1.0	Virtualization MO	DM WG	1.0	VirMO-V1_0.ddf	OMA-ER-VirMO-V1_0
urn:oma:mo:oma-vmimgmo:1.0	Virtual Machine Image MO	DM WG	1.0	VMImgMO-V1_0.ddf	OMA-ER-VirMO-V1_0

Appendix A. Change History (Informative)

A.1 Approved Version History

Reference	Date	Description
n/a	n/a	No prior version

A.2 Draft/Candidate Version 1.0 History

Document Identifier	Date	Sections	Description
Draft Versions OMA-ER-VirMO-V1_0	18 Dec 2011	All	First draft baseline as agreed in OMA-DM-VirMO-2011-0001-INP_ER_Baseline
	11 Apr 2012	3.2, 5.1, Appendix B	Incorporates CRs: OMA-DM-VirMO-2012-0002R02-CR_Controlling_Data_Sharing OMA-DM-VirMO-2012-0003R02-CR_Support_existing_MOs OMA-DM-VirMO-2012-0004R02-CR_Provisioning_Firmware OMA-DM-VirMO-2012-0006R01-CR_Provisioning_of_a_Secure_Enterprise_Domain_Use_Case OMA-DM-VirMO-2012-0007R01-CR_Employee_Leaving_Use_Case OMA-DM-VirMO-2012-0009-CR_Peripheral_Access_Use_Case.doc OMA-DM-VirMO-2012-0014-CR_VM_Definition OMA-DM-VirMO-2012-0015-CR_Virtualization_enabled_device_def.doc Deleted some template comments.
	10 May 2012	1, 2, 3.2, 5.1, Appendix B	Incorporates CRs: OMA-DM-VirMO-2012-0012R02-CR_Requirements OMA-DM-VirMO-2012-0016R01-CR_VM_ID_and_VM_Status OMA-DM-VirMO-2012-0017R03-CR_Usecase_Req_Def_for_Lock_Unlock OMA-DM-VirMO-2012-0018R01-CR_Suspend_Resume_Usecase_Req OMA-DM-VirMO-2012-0019R02-CR_Scope_and_Reference OMA-DM-VirMO-2012-0020R01-CR_Provisioning_Unlock_Authentication_Data_REQ
	23 May 2012	4	Incorporates CRs: OMA-DM-VirMO-2012-0021R01-CR_Introduction_Chapter
	28 Aug 2012	4, 6	Incorporates CRs: OMA-DM-VirMO-2012-0022-CR_Introduction_Clarification OMA-DM-VirMO-2012-0023R02-CR_AD_baseline
	10 Oct 2012	7, 8, 9	Incorporates CRs: OMA-DM-VirMO-2012-0024R03-CR_State_Machine_and_Primitives OMA-DM-VirMO-2012-0025R03-CR_Baseline_MO OMA-DM-VirMO-2012-0026R02-CR_Creating_Deleting_VM OMA-DM-VirMO-2012-0029-CR_Creating_Deleting_VM_Clarifications OMA-DM-VirMO-2012-0030-CR_State_Values
	21 Nov 2012	5.1, 8, 9, 10.1, 10.2	Incorporates CRs: OMA-DM-VirMO-2012-0031R02-CR_CreateParams OMA-DM-VirMO-2012-0032R01-CR_VMImgMO OMA-DM-VirMO-2012-0033-CR_VM_Img_Delivery_Proc OMA-DM-VirMO-2012-0034-CR_UpdateVMReq Editorial changes
	13 Feb 2013	11	Incorporated CRs: OMA-DM-VirMO-2013-0001R01-CR_Device_Management_for_VM

Document Identifier	Date	Sections	Description
	20 Feb 2013	All	Incorporated CRs: OMA-DM-VirMO-2013-0004-CR_Editorials_and_Bug_Fixs OMA-DM-VirMO-2013-0005R01-CR_VM_Image_Deletion_after_creating_VM OMA-DM-VirMO-2013-0002R02-CR_Resource_Management Editorial changes
	14 Apr 2013	11	Incorporated CRs: OMA-DM-VirMO-2013-0006R01-CR_VirMO_x_DMTree_node OMA-DM-VirMO-2013-0008-CR_Using_DCMO OMA-DM-VirMO-2013-0009-CR_UI_for_VM
	1 May 2013	3.2, 5, 11.1, 12, 13, Appendix C	Incorporated CRs: OMA-DM-VirMO-2013-0007-CR_VM_Bootstrapping OMA-DM-VirMO-2013-0010-CR_Image_Ready_Alert OMA-DM-VirMO-2013-0011-CR_SCR_Table OMA-DM-VirMO-2013-0012R01-CR_REQ_CleanUp
	3 May 2013	2.1, 13.1	Sorting of references in alphabetical order Update of dates in the supporting files document listing Editorial changes
	5 Jun	All	Incorporated CRs: OMA-DM-VirMO-2013-0016R02-CR_Resolving_CONR_Comments Sorting of references in alphabetical order
Candidate Version OMA-ER-VirMO-V1_0	25 Jun 2013	n/a	Status changed to Candidated by TP TP Ref # OMA-TP-2013-0205-INP_Virtualisation_MO_V1_0_ERP_and_ETR_for_Candidate_Approval

Appendix B. Use Cases (Informative)

B.1 Provisioning of a Secure Enterprise Virtual Machine

Today, it is a common practice for employees to keep confidential corporate data, such as documents, presentations, e-mail, calendar, contacts, and customer data, on their privately owned devices. This amalgamation of personal and enterprise usage is problematic. There is a risk that confidential corporate data fall into the wrong hands if the device is stolen, or the user unintentionally installs a malware-crippled app. In addition, the end user's personal data is at risk of being wiped from the device by the administrator of the enterprise e-mail server.

By provisioning a Virtual Machine dedicated to enterprise apps and data, there will be a secure separation between the personal usage and enterprise usage in the single device.

B.1.1 Short Description

An employee brings her personal wireless device to the enterprise. The enterprise administrator needs to give the new employee access to enterprise apps such as corporate e-mail from a secure environment on the employee's personal device. The enterprise administrator provisions an enterprise Virtual Machine to the virtualization-enabled device where the enterprise apps run and the enterprise data is stored.

B.1.2 Market benefits

This will give enterprises the benefit of letting employees bring their own devices into the organization, without having to expose the confidential enterprise data to apps on the employee’s personal device.

Also, employees will not need to worry about having their personal data deleted by the administrator of the enterprise e-mail server.

B.2 Deleting the Enterprise Virtual Machine when Employee Leaving the Employment

By provisioning a Virtual Machine dedicated to enterprise apps and data, there will be a secure separation between the personal usage and enterprise usage in the single device. When an employee leaves her employment, there is a need to remove the enterprise Virtual Machine from the device.

B.2.1 Short Description

The employee, with a personal device provisioned with an enterprise Virtual Machine, terminates her employment. The enterprise administrator removes the enterprise apps and data from the device by deleting the enterprise Virtual Machine from the employee’s device.

B.2.2 Market benefits

This will give enterprises the benefit of efficiently preventing employees leaving the organization from having continued access to enterprise data stored on the device.

Also, employees will be able to keep using the device for personal use with all personal data intact, even after the enterprise Virtual Machine has been removed.

B.3 Control of Peripheral Access on the Enterprise Virtual Machine

By provisioning a Virtual Machine dedicated to enterprise apps and data, there will be a secure separation between the personal usage and enterprise usage in the single device. High-security organizations need to block access to peripheral devices from the enterprise Virtual Machine.

B.3.1 Short Description

The organization, with a strong focus on security, needs to prevent transfer of corporate information and side loading of apps into the enterprise Virtual Machine. They accomplish this by not allowing USB connected storage to the device, as well as Wi-Fi, Bluetooth or NFC connections for the enterprise Virtual Machine.

B.3.2 Market benefits

This will give enterprises the benefit of efficiently preventing unauthorized copies of corporate information from the device, as well as maintaining a secure environment on the device.

B.4 Suspending and Resuming the Virtual Machine

B.4.1 Short Description

The DiagnosticKing, a device diagnostic company, is requested to remotely investigate one of customer's devices. The reported problems were the time displayed on the screen is incorrect all the time. When the DiagnosticKing connects to the device, it finds a Virtual Machine running which largely consumes resources, and disturbs the smooth investigations. The DiagnosticKing suspends the Virtual Machine, and keeps going for the diagnostic procedures. After finding and fixing the problem, the DiagnosicKing resumes the Virtual Machine continuing from the saved states. With this suspend and resume feature, no data and states of the Virtual Machine are lost from the customer's perspective, and the DiagnosticKing earns the resources for high priority jobs.

B.4.2 Market benefits

The suspend and resume feature of the Virtual Machine gives the operational flexibilities for users, service center staffs and others when the execution of the Virtual Machine needs to be stopped, and later continues from the saved states.

Appendix C. Static Conformance Requirements (Normative)

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

C.1 ERDEF for Virtualization MO - Client Requirements

This section is normative.

Item	Feature / Application	Requirement
OMA-ERDEF-VirMO-C-001-M	VirMO Client	

Table 2: ERDEF for Virtualization MO Client-side Requirements

C.2 ERDEF for Virtualization MO - Server Requirements

This section is normative.

Item	Feature / Application	Requirement
OMA-ERDEF-VirMO-S-001-M	VirMO Server	

Table 3: ERDEF for Virtualization MO Server-side Requirements

C.3 SCR for Management Object Tree Structure

Item	Function	Reference	Requirement
VirMO-T-001-M	Use of appropriate Management Object identifier for Management Objects	Section 8, 9	
VirMO-T-002-M	Support for Required nodes under the root node	Section 8, 9	
VirMO-T-003-O	Support for Optional nodes	Section 8, 9	
VirMO-T-004-M	Support for Required nodes under an Optional node if the Optional node is supported	Section 8, 9	

C.4 SCR for Virtualization MO Client

C.4.1 SCR for Virtualization MO

Item	Function	Reference	Requirement
VirMO-C-001-M	Support for assigning the VMID when the sub-tree under the *VirMO* node is created	Section 8, 10.2	
VirMO-C-002-M	Support for updating the *VirMO/<x>/State* node when the state of the Virtual Machine is changed	Section 8, 10.2	
VirMO-C-003-M	Support for creating the VM according to the *VirMO/<x>/CreatePara*	Section 8, 10.2	

Item	Function	Reference	Requirement
	ms node		
VirMO-C-004-M	Support for deleting the VM	Section 8, 10.3	
VirMO-C-005-M	Support for suspending the VM	Section 8	
VirMO-C-006-M	Support for resuming the suspended VM	Section 8	
VirMO-C-007-M	Support for powering on the VM	Section 8	
VirMO-C-008-M	Support for powering off the VM	Section 8	
VirMO-C-009-O	Support for managing the DM Tree for the VM under *VirMO/<x>/DMTree* node	Section 8, 11.2	
VirMO-C-010-O	Support for allocating device resources for the VM	Section 8	

C.4.2 SCR for Virtual Machine Image MO

Item	Function	Reference	Requirement
VirMO-C-011-M	Support for delivering the VM image	Section 9, 10.1	
VirMO-C-012-M	Support for deleting the VM image	Section 9	

C.5 SCR for Virtualization MO Server

Item	Function	Reference	Requirement
VirMO-S-001-M	Support for the Virtualization MO and Virtual Machine Image MO	Section 8, 9	