(12) 특허협력조약에 의하여 공개된 국제출원

(19) 세계지식재산권기구
국제사무국

(43) 국제공개일
2013 년 8 월 8 일 (08.08.2013)

WIPO | PCT

(10) 국제공개번호
WO 2013/115565 A2

(51) 국제특허분류:
G06F 9/44 (2006.01) *G06F 15/16* (2006.01)

(21) 국제출원번호: PCT/KR2013/000763

(22) 국제출원일: 2013 년 1 월 30 일 (30.01.2013)

(25) 출원언어: 한국어

(26) 공개언어: 한국어

(30) 우선권정보:

61/592,555	2012 년 1 월 30 일 (30.01.2012)	US
61/620,444	2012 년 4 월 5 일 (05.04.2012)	US
61/696,310	2012 년 9 월 4 일 (04.09.2012)	US
61/700,349	2012 년 9 월 13 일 (13.09.2012)	US
61/716,625	2012 년 10 월 22 일 (22.10.2012)	US
61/738,390	2012 년 12 월 17 일 (17.12.2012)	US

(71) 출원인: **엘지전자 주식회사 (LG ELECTRONICS INC.)** [KR/KR]; 150-721 서울시 영등포구 여의도동 20, Seoul (KR).

(72) 발명자: **박승규 (PARK, Seungkyu)**; 431-080 경기도 안양시 동안구 호계 1 동 533 번지 엘지전자 특허센터, Gyeonggi-do (KR). **김성윤 (KIM, Seongyun)**; 431-080 경기도 안양시 동안구 호계 1 동 533 번지 엘지전자 특허센터, Gyeonggi-do (KR).

(74) 대리인: **김용인 (KIM, Yong In)** 등; 138-861 서울시 송파구 잠실동 175-9 현대빌딩 7 층 KBK 특허법률사무소, Seoul (KR).

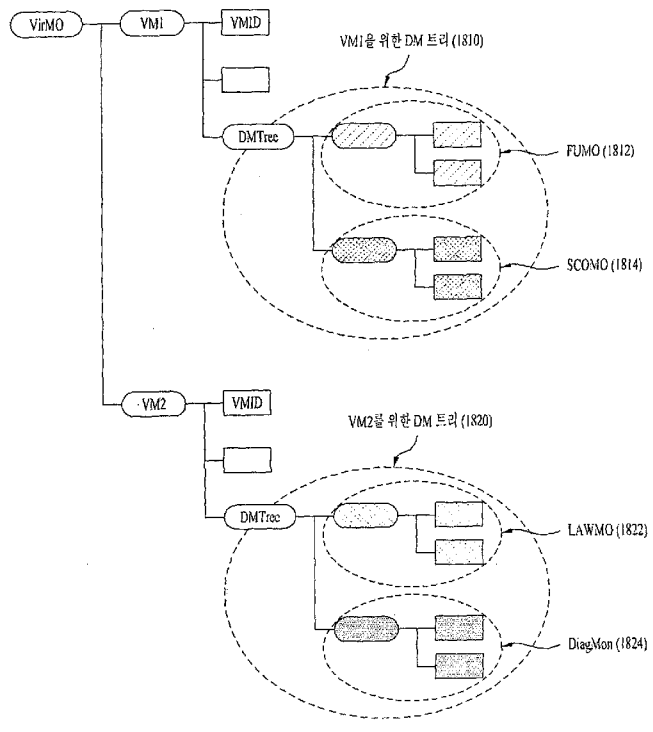
(81) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 국내 권리의 보호를 위하여): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) 지정국 (별도의 표시가 없는 한, 가능한 모든 종류의 역내 권리의 보호를 위하여): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), 유라시아 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 유럽 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC,

[다음 쪽 계속]

(54) **Title**: METHOD FOR MANAGING VIRTUAL MACHINE AND DEVICE THEREFOR

(54) **발명의 명칭** : 가상 머신 관리 방법 및 이를 위한 장치



1810... DM tree for VM1
1820... DM tree for VM2

(57) **Abstract**: The present invention relates to a virtual machine, and more particularly, to a method for performing device management for a virtual machine in a terminal comprising a plurality of virtual machines, and a device therefor, the method comprising the steps of: generating a specific virtual machine; constituting at least one management object, which is required for providing device management for the specific virtual machine, in a virtualization management object; receiving a device management command from a server; checking whether the device management command is for the specific virtual machine; and processing the device management command if the device management command is for the specific virtual machine, wherein the virtualization management object includes a first node for setting up information related to the specific virtual machine and includes, below the first node, a second node for setting up information required for device management for the specific virtual machine, and the at least one management object is formed below the second node.

(57) 요약서:

[다음 쪽 계속]

WO 2013/115565 A2

MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

공개:

— 국제조사보고서 없이 공개하며 보고서 접수 후 이를 별도 공개함 (규칙 48.2(g))

본 발명은 가상 머신에 관한 것이다. 구체적으로, 본 발명은 복수의 가상 머신을 포함하는 단말에서 가상 머신을 위한 장치 관리를 수행하는 방법 및 이를 위한 장치에 관한 것이며, 상기 방법은 특정 가상 머신을 생성하는 단계; 가상화 관리 오브젝트(virtualization management objection)에 상기 특정 가상 머신을 위한 장치 관리를 제공하는 데 필요한 적어도 하나의 관리 오브젝트를 구성하는 단계; 서버로부터 장치 관리 명령을 수신하는 단계; 상기 장치 관리 명령이 상기 특정 가상 머신을 위한 장치 관리 명령인지 여부를 확인하는 단계; 및 상기 장치 관리 명령이 상기 특정 가상 머신을 위한 장치 관리 명령인 경우 상기 장치 관리 명령을 처리하는 단계를 포함하되, 상기 가상화 관리 오브젝트는 상기 특정 가상 머신과 관련된 정보를 설정하기 위한 제 1 노드를 포함하고, 상기 제 1 노드 아래에 상기 특정 가상 머신을 위한 장치 관리에 필요한 정보를 설정하기 위한 제 2 노드를 포함하며, 상기 적어도 하나의 관리 오브젝트는 상기 제 2 노드 아래에 구성될 수 있다.

명세서

발명의 명칭: 가상 머신 관리 방법 및 이를 위한 장치

기술분야

[1] 본 발명은 가상 머신에 관한 것으로서, 구체적으로 서버에서 단말의 가상 머신을 관리하는 방법 및 이를 위한 장치에 관한 것이다.

배경기술

[2] 종래 관리 모델은 단일 컴퓨팅 환경(computing environment)에 하나의 운영체제(Operating System, OS)를 설치하고 OS를 통해 웹 서비스와 같은 서비스들을 주로 제공한다. 예를 들어, 데이터베이스 서비스를 제공하는 경우, 새로운 컴퓨팅 환경에 OS 및 데이터베이스 프로그램을 설치하여 서비스를 제공한다. 일반적으로, 종래 관리 모델은 단일 플랫폼 상에서 서로 다른 서비스들을 운영하지 않는다. 따라서, 새로운 서비스를 제공하기 위해서는 서로 다른 플랫폼이 요구되며, 이로 인해 설치 및 관리 비용이 증가될 수 있다.

[3] 이러한 문제점들을 극복하기 위해 가상화(virtualization) 기술이 사용된다. 가상화는 하드웨어 플랫폼, OS, 디바이스, 또는 네트워크 등의 가상 버전(virtual version)을 제공하는 기술이다. 일 예로, OS 가상화는 OS와 물리적 디바이스 간에 직접 통신을 제공하는 대신에 하드웨어 플랫폼을 추상화하여 추상화된 하드웨어 플랫폼(abstract hardware platform)을 제공한다. 사용자는 추상화된 하드웨어 플랫폼 상에 별도의 컴퓨팅 환경을 가지는 OS를 설치할 수 있다. 가상화를 통해 단일 플랫폼 상에서 복수의 독립적인 컴퓨팅 환경을 제공할 수 있으며 단일 플랫폼 상에서 각 독립적인 컴퓨팅 환경을 제어 프로그램을 통해 관리할 수 있으므로 가상화는 장점이 있다. 또한, 가상화를 통해 하나의 하드웨어 플랫폼 상에서 복수의 서비스를 운영할 수 있기 때문에 물리적 엔티티들의 개수를 감소시킬 수 있으므로 비용적인 측면에서도 유리하다.

[4] 추상화된 하드웨어 플랫폼을 제공하는 가상화 기술로서 가상 머신이 있으며, PC(Personal Computer) 환경에서 VM웨어라는 제품이 널리 알려져 있다. 가상 머신은 독자적인 실행 환경을 제공해 주는데, 독자적인 실행 환경이란 하나의 가상 머신에서 동작하는 애플리케이션들이 다른 가상 머신에서 동작하는 애플리케이션들에 영향을 주지 않는 것을 의미한다.

[5] VM웨어는 PC(Personal Computing) 환경에서의 가상화 기술로 널리 사용되고 있지만, VM웨어를 단말에서 서버 중심의 VM 관리 기술로 적용하기에는 무리가 있다. VM웨어는 기본적으로 기기 안에서 추상화된 하드웨어 플랫폼을 위한 기술로 개발 되었지만 원격의 서버가 VM을 관리할 수 있는 기술을 포함하고 있지 않다. 따라서 원격의 서버가 단말에 가상 머신을 생성하고 제거하는 기본적인 관리 기술에서부터 새로운 접근이 필요하다.

[6] 뿐만 아니라, VM웨어는 원격의 서버가 단말에 생성된 가상 머신을 잠금(lock),

잠금해제(unlock), 업데이트(update)하는 것과 같이 원격 제어할 수 있는 방안이 요구된다.

발명의 상세한 설명

기술적 과제

[7] 본 발명은 단말에서 가상 머신을 효율적으로 관리하는 방법을 제공하는 데 있다. 또한, 본 발명은 단말에서 복수의 가상 머신을 운용하는 경우 서버가 단말의 각 가상 머신을 효율적으로 관리하는 방법을 제공하는 데 있다.

[8] 또한, 본 발명은 서버가 단말에서 새로운 가상 머신을 효율적으로 생성하고 제거하는 방법을 제공하는 데 있다.

[9] 또한, 본 발명은 서버가 가상 머신 이미지를 단말로 전달하는 방법을 제공하고, 또한 서버가 전달된 가상 머신 이미지를 이용하여 가상 머신을 효율적으로 생성하는 방법을 제공하는 데 있다.

[10] 또한, 본 발명은 서버가 가상 머신을 복제(duplicate)하는 방법을 제공하는 데 있다.

[11] 또한, 본 발명은 서버가 가상 머신을 잠금 및 잠금해제 하는 방법을 제공하는 데 있다. 또한, 본 발명은 효율적인 사용자 잠금해제 방법을 제공하는 데 있다.

[12] 또한, 본 발명은 서버가 OMA DM을 이용하여 가상 머신을 위한 장치 관리(Device Management for Virtual Machine)를 가능하게 하는 방법을 제공하는 데 있다.

[13] 본 발명에서 이루고자 하는 기술적 과제들은 상기 기술적 과제로 제한되지 않으며, 언급하지 않은 또 다른 기술적 과제들은 아래의 기재로부터 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 명확하게 이해될 수 있을 것이다.

과제 해결 수단

[14] 본 발명의 일 양상으로, 복수의 가상 머신을 포함하는 단말에서 가상 머신을 위한 장치 관리를 수행하는 방법이 제공되며, 상기 방법은 특정 가상 머신을 생성하는 단계; 가상화 관리 오브젝트(virtualization management objection)에 상기 특정 가상 머신을 위한 장치 관리를 제공하는 데 필요한 적어도 하나의 관리 오브젝트를 구성하는 단계; 서버로부터 장치 관리 명령을 수신하는 단계; 상기 장치 관리 명령이 상기 특정 가상 머신을 위한 장치 관리 명령인지 여부를 확인하는 단계; 및 상기 장치 관리 명령이 상기 특정 가상 머신을 위한 장치 관리 명령인 경우 상기 장치 관리 명령을 처리하는 단계를 포함하되, 상기 가상화 관리 오브젝트는 상기 특정 가상 머신과 관련된 정보를 설정하기 위한 제1 노드를 포함하고, 상기 제1 노드 아래에 상기 특정 가상 머신을 위한 장치 관리에 필요한 정보를 설정하기 위한 제2 노드를 포함하며, 상기 적어도 하나의 관리 오브젝트는 상기 제2 노드 아래에 구성될 수 있다.

[15] 바람직하게는, 상기 가상화 관리 오브젝트는 상기 제1 노드 아래에 상기 특정

가상 머신에 대한 식별 정보를 설정하기 위한 노드를 더 포함하고, 상기 확인하는 단계는 상기 장치 관리 명령이 상기 제2 노드 아래에 위치한 노드를 타겟으로 하는 경우 상기 장치 관리 명령은 상기 식별 정보에 의해 식별되는 가상 머신을 위한 것으로 결정하는 것을 포함할 수 있다.

[16] 바람직하게는, 상기 적어도 하나의 관리 오브젝트는 상기 제2 노드 아래에 구성되는 루트 노드를 가지고, 상기 루트 노드는 관리 오브젝트 식별 정보를 포함하며, 상기 적어도 하나의 관리 오브젝트는 상기 관리 오브젝트 식별 정보에 의해 구별될 수 있다.

[17] 바람직하게는, 상기 가상화 관리 오브젝트는 단말의 장치 관리 트리에 포함될 수 있다.

[18] 바람직하게는, 상기 특정 가상 머신에 대한 식별 정보는 단말에 의해 주어지고 상기 가상화 관리 오브젝트 내에서 상기 특정 가상 머신을 고유하게 지시할 수 있다.

[19] 바람직하게는, 상기 제1 노드는 VirMO/<x> 노드이고, 상기 제2 노드는 VirMO/<x>/DMTree 노드이며, <x>는 상기 특정 가상 머신을 지시하는 정보일 수 있다.

[20] 바람직하게는, 상기 적어도 하나의 관리 오브젝트는 FUMO(Firmware Update Management Object), SCOMO(Software Component Management Object), 및/또는 LAWMO(Lock and Wipe Management Object)를 포함할 수 있다.

[21] 바람직하게는, 상기 가상화 관리 오브젝트는 상기 제1 노드 아래에 사용자 상호 작용에 관한 정보를 설정하기 위한 제3 노드와 사용자 상호 작용에 관한 명령을 수신하기 위한 제4 노드를 포함하며, 상기 장치 관리 명령을 수신하는 단계는 상기 제4 노드를 통해 실행 명령을 수신하는 것을 포함하고, 상기 장치 관리 명령을 처리하는 단계는 상기 제3 노드에 설정된 상기 사용자 상호 작용에 관한 명령 정보를 실행하는 것을 포함할 수 있다.

[22] 바람직하게는, 상기 제3 노드는 VirMO/<x>/UI/UICommands 노드이고, 상기 제4 노드는 VirMO/<x>/Operations/ShowUI 노드이며, <x>는 상기 특정 가상 머신을 지시하는 정보일 수 있다.

[23] 본 발명의 다른 양상으로, 복수의 가상 머신을 포함하고 가상 머신을 위한 장치 관리를 수행하는 단말이 제공되며, 상기 단말은 프로세서; 및 송수신 모듈을 포함하며, 상기 프로세서는 특정 가상 머신을 생성하도록 구성되고, 가상화 관리 오브젝트(virtualization management objection)에 상기 특정 가상 머신을 위한 장치 관리를 제공하는 데 필요한 적어도 하나의 관리 오브젝트를 구성하도록 구성되고, 서버로부터 장치 관리 명령을 수신하도록 구성되고, 상기 장치 관리 명령이 상기 특정 가상 머신을 위한 장치 관리 명령인지 여부를 확인하도록 구성되고, 상기 장치 관리 명령이 상기 특정 가상 머신을 위한 장치 관리 명령인 경우 상기 장치 관리 명령을 처리하도록 구성되며, 상기 가상화 관리 오브젝트는 상기 특정 가상 머신과 관련된 정보를 설정하기 위한 제1 노드를 포함하고, 상기

제1 노드 아래에 상기 특정 가상 머신을 위한 장치 관리에 필요한 정보를 설정하기 위한 제2 노드를 포함하며, 상기 적어도 하나의 관리 오브젝트는 상기 제2 노드 아래에 구성될 수 있다.

[24] 바람직하게는, 상기 가상화 관리 오브젝트는 상기 제1 노드 아래에 상기 특정 가상 머신에 대한 식별 정보를 설정하기 위한 노드를 더 포함하고, 상기 확인하는 것은 상기 장치 관리 명령이 상기 제2 노드 아래에 위치한 노드를 타겟으로 하는 경우 상기 장치 관리 명령은 상기 식별 정보에 의해 식별되는 가상 머신을 위한 것으로 결정하는 것을 포함할 수 있다.

[25] 바람직하게는, 상기 적어도 하나의 관리 오브젝트는 상기 제2 노드 아래에 구성되는 루트 노드를 가지고, 상기 루트 노드는 관리 오브젝트 식별 정보를 포함하며, 상기 적어도 하나의 관리 오브젝트는 상기 관리 오브젝트 식별 정보에 의해 구별될 수 있다.

[26] 바람직하게는, 상기 가상화 관리 오브젝트는 단말의 장치 관리 트리에 포함될 수 있다.

[27] 바람직하게는, 상기 특정 가상 머신에 대한 식별 정보는 단말에 의해 주어지고 상기 가상화 관리 오브젝트 내에서 상기 특정 가상 머신을 고유하게 지시할 수 있다.

[28] 바람직하게는, 상기 제1 노드는 VirMO/<x> 노드이고, 상기 제2 노드는 VirMO/<x>/DMTree 노드이며, <x>는 상기 특정 가상 머신을 지시하는 정보일 수 있다.

[29] 바람직하게는, 상기 적어도 하나의 관리 오브젝트는 FUMO(Firmware Update Management Object), SCOMO(Software Component Management Object), 및/또는 LAWMO(Lock and Wipe Management Object)를 포함할 수 있다.

[30] 바람직하게는, 상기 가상화 관리 오브젝트는 상기 제1 노드 아래에 사용자 상호 작용에 관한 정보를 설정하기 위한 제3 노드와 사용자 상호 작용에 관한 명령을 수신하기 위한 제4 노드를 포함하며, 상기 장치 관리 명령을 수신하는 것은 상기 제4 노드를 통해 실행 명령을 수신하는 것을 포함하고, 상기 장치 관리 명령을 처리하는 것은 상기 제3 노드에 설정된 상기 사용자 상호 작용에 관한 명령 정보를 실행하는 것을 포함할 수 있다.

[31] 바람직하게는, 상기 제3 노드는 VirMO/<x>/UI/UICommands 노드이고, 상기 제4 노드는 VirMO/<x>/Operations/ShowUI 노드이며, <x>는 상기 특정 가상 머신을 지시하는 정보일 수 있다.

[32] 본 발명의 또 다른 양상으로, 가상화 관리 오브젝트와 가상 머신 이미지 관리 오브젝트를 가지는 단말에서 가상 머신을 생성하는 방법이 제공되고, 상기 가상 머신 이미지 관리 오브젝트는 복수의 가상 머신들의 가상 머신 이미지 식별 정보가 각각 설정된 복수의 노드들을 포함하며, 상기 방법은 서버로부터 실행 명령을 수신하는 단계; 및 상기 실행 명령이 상기 가상화 관리 오브젝트의 가상 머신 생성 동작과 관련된 제1 노드로 전송된 경우, 상기 가상화 관리 오브젝트의

제2 노드에 설정된 가상 머신 생성 파라미터에 따라 특정 가상 머신을 생성하는 단계를 포함하되, 상기 제2 노드에 설정된 가상 머신 생성 파라미터는 상기 특정 가상 머신을 생성하는 데 사용되는 가상 머신 이미지를 특정하는 제1 정보를 포함하고, 상기 제1 정보는 상기 복수의 노드들에 설정된 가상 머신 이미지 식별 정보들 중 하나일 수 있다.

[33] 바람직하게는, 상기 가상 머신 생성 파라미터는 상기 생성된 특정 가상 머신의 상태를 지시하는 제2 정보를 더 포함하고, 상기 제2 정보가 제1 값인 경우, 상기 생성된 특정 가상 머신은 실행 중인 상태에 있고, 상기 제2 정보가 제2 값인 경우, 상기 생성된 특정 가상 머신은 파워 오프 상태에 있을 수 있다.

[34] 바람직하게는, 상기 가상 머신 이미지 관리 오브젝트는 단말의 장치 관리 트리에 포함될 수 있다.

[35] 바람직하게는, 상기 가상 머신 이미지 식별 정보는 단말에 의해 주어지고 상기 가상 머신 이미지 관리 오브젝트 내에서 상기 특정 가상 머신 이미지를 고유하게 지시할 수 있다.

[36] 바람직하게는, 상기 방법은 동기식 보고 메커니즘 또는 비동기식 보고 메커니즘을 이용하여 상기 서버에 응답하는 단계를 더 포함할 수 있다.

[37] 바람직하게는, 상기 방법은 상기 서버로부터 실행 명령을 수신하는 단계를 더 포함하고, 상기 실행 명령이 상기 가상화 관리 오브젝트의 가상 머신 제거 동작과 관련된 제1 노드로 전송된 경우, 상기 생성된 특정 가상 머신은 제거될 수 있다.

[38] 본 발명의 또 다른 양상으로, 가상화 관리 오브젝트와 가상 머신 이미지 관리 오브젝트를 가지며 가상 머신을 생성하는 단말이 제공되고, 상기 가상 머신 이미지 관리 오브젝트는 복수의 가상 머신들의 가상 머신 이미지 식별 정보가 각각 설정된 복수의 노드들을 포함하며, 상기 단말은 프로세서; 및 송수신 모듈을 포함하고, 상기 프로세서는 상기 송수신 모듈을 통해 서버로부터 실행 명령을 수신하도록 구성되고, 상기 실행 명령이 상기 가상화 관리 오브젝트의 가상 머신 생성 동작과 관련된 제1 노드로 전송된 경우, 상기 가상화 관리 오브젝트의 제2 노드에 설정된 가상 머신 생성 파라미터에 따라 특정 가상 머신을 생성하도록 구성되며, 상기 제2 노드에 설정된 가상 머신 생성 파라미터는 상기 특정 가상 머신을 생성하는 데 사용되는 가상 머신 이미지를 특정하는 제1 정보를 포함하고, 상기 제1 정보는 상기 복수의 노드들에 설정된 가상 머신 이미지 식별 정보들 중 하나일 수 있다.

[39] 바람직하게는, 상기 가상 머신 생성 파라미터는 상기 생성된 특정 가상 머신의 상태를 지시하는 제2 정보를 더 포함하고, 상기 제2 정보가 제1 값인 경우, 상기 생성된 특정 가상 머신은 실행 중인 상태에 있고, 상기 제2 정보가 제2 값인 경우, 상기 생성된 특정 가상 머신은 파워 오프 상태에 있을 수 있다.

[40] 바람직하게는, 상기 가상 머신 이미지 관리 오브젝트는 단말의 장치 관리 트리에 포함될 수 있다.

[41] 바람직하게는, 상기 가상 머신 이미지 식별 정보는 단말에 의해 주어지고 상기 가상 머신 이미지 관리 오브젝트 내에서 상기 특정 가상 머신 이미지를 고유하게 지시할 수 있다.

[42] 바람직하게는, 상기 프로세서는 또한 동기식 보고 메커니즘 또는 비동기식 보고 메커니즘을 이용하여 상기 서버에 응답하도록 구성될 수 있다.

[43] 바람직하게는, 상기 프로세서는 또한 상기 서버로부터 실행 명령을 수신하도록 구성되고, 상기 실행 명령이 상기 가상화 관리 오브젝트의 가상 머신 제거 동작과 관련된 제1 노드로 전송된 경우, 상기 생성된 특정 가상 머신은 제거될 수 있다.

[44] 본 발명에 의하면, 서버는 단말의 가상 머신을 효율적으로 관리할 수 있다. 또한, 본 발명에 의하면, 단말에서 복수의 가상 머신을 운용하는 경우에도 서버는 단말의 각 가상 머신을 효율적으로 관리할 수 있다.

[45] 또한, 본 발명에 의하면, 서버는 단말에 새로운 가상 머신을 효율적으로 생성하고 제거할 수 있다.

[46] 또한, 본 발명에 의하면, 서버는 가상 머신 이미지를 단말로 효율적으로 전달할 수 있고, 전달된 가상 머신 이미지를 이용하여 가상 머신을 효율적으로 생성할 수 있다. 구체적으로, 서버는 단말에 가상 머신 생성에 필요한 VM 이미지를 전달할 수 있고, VM 이미지 타입에 따라 VM 생성을 제어할 수 있다. 이로써 서버는 VM 이미지를 한번 단말에 전달하여, 동일한 VM 이미지로 다수의 VM을 단말에 생성할 수도 있고, 하나의 VM 이미지가 하나의 VM 생성에만 이용되도록 할 수 있다

[47] 또한, 본 발명에 의하면, 서버는 단말에서 가상 머신을 복제(duplicate)할 수 있다.

[48] 또한, 본 발명에 의하면, 서버는 가상 머신을 잠금 및 잠금해제 할 수 있다. 또한, 본 발명에 의하면 효율적인 사용자 잠금해제가 가능하다. 구체적으로, 서버는 원격에서 기밀성 데이터를 저장하고 있는 가상 머신을 잠글 수 있어 가상 머신안의 데이터를 안전하게 보호할 수 있다. 서버는 또한 잠긴 가상 머신을 잠금해제할 수도 있다. 가상 머신 잠금은 가상 머신 내부 상태와 무관하게 이루어 질 수 있어 가상 머신 관리가 효율적일 수 있다. 또한, 서버는 잠긴 가상 머신에 대해서 사용자가 직접 가상 머신을 안전하고 편리하게 잠금해제하게 할 수 있다.

[49] 또한, 본 발명에 의하면, 서버는 OMA DM을 이용하여 가상 머신을 위한 장치 관리(Device Management for Virtual Machine)를 수행할 수 있다. 구체적으로, 서버는 생성된 가상 머신에 대해서 새로운 DM 클라이언트를 생성함이 없이 장치 관리(Device Management) 기능을 제공할 수 있다. 가상 머신 장치 관리(Virtual Machine Device Management)를 위해 서버는 가상 머신을 위해 필요한 최소한의 관리 오브젝트(MO) 만을 생성하면 된다.

[50] 본 발명에서 얻을 수 있는 효과는 이상에서 언급한 효과들로 제한되지 않으며,

언급하지 않은 또 다른 효과들은 아래의 기재로부터 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 명확하게 이해될 수 있을 것이다.

도면의 간단한 설명

[51] 첨부 도면은 본 발명에 관한 이해를 돕기 위해 상세한 설명의 일부로 포함되는 것으로서, 본 발명에 대한 실시예를 제공하고 상세한 설명과 함께 본 발명의 기술적 사상을 설명한다.

[52] 도 1은 하나의 디바이스 상에서 구성(configure)될 수 있는 가상 머신(VM)들을 예시한 것이다.

[53] 도 2는 OMA DM 아키텍처를 기반으로 한 가상화 관리 오브젝트(Virtualization Management Object) 아키텍처를 예시한 것이다.

[54] 도 3은 본 발명에 적용될 수 있는 가상화 타입을 예시한 것이다.

[55] 도 4는 타입 I 가상화를 통해 가상화 관리 오브젝트를 위한 엔티티들이 구현되는 예를 예시한다.

[56] 도 5는 타입 II 가상화를 통해 가상화 관리 오브젝트를 위한 엔티티들이 구현되는 예를 예시한다.

[57] 도 6은 VM의 가능한 상태와 프리머티브(primitive)를 예시하는 상태 머신(state machine)의 예를 도시한 것이다.

[58] 도 7은 추가적인 VM의 가능한 상태와 프리머티브를 포함하는 상태 머신의 예를 예시한 것이다.

[59] 도 8은 VM 관리를 위한 가상화 관리 오브젝트의 예를 예시한 것이다.

[60] 도 9는 가상 머신 이미지 관리 오브젝트(Virtual Machine Image Management Object)를 예시한 것이다.

[61] 도 10은 VirMO 서버가 VM 이미지를 디바이스에 전달(deliver)하는 절차를 예시한 것이다.

[62] 도 11은 VirMO 서버가 디바이스에 가상 머신을 생성하는 절차를 예시한 것이다.

[63] 도 12는 VirMO 서버가 디바이스의 가상 머신을 제거하는 절차를 예시한 것이다.

[64] 도 13과 도 14는 사용자가 VM을 잠금해제할 때 VirMO 서버와 VirMO 클라이언트에서 각각 수행되는 절차를 예시한 것이다.

[65] 도 15와 도 16은 별도의 DM 클라이언트를 이용하여 장치 관리를 수행하는 방법의 예를 예시한 것이다.

[66] 도 17은 공유 DM 클라이언트를 이용하여 장치 관리를 수행하는 방법의 예를 예시한 것이다.

[67] 도 18은 VM 장치 관리(Device Management for Virtual Machine)를 위해 각 VM을 위한 DM 트리를 구성하는 방법을 예시한 것이다.

[68] 도 19는 VM 장치 관리를 수행하는 절차의 순서도를 예시한 것이다.

[69] 도 20은 VM에서 발생된 일반 경보(Generic Alert)를 VirMO 서버로 보내고 처리하는 절차의 순서도를 예시한 것이다.

[70] 도 21은 소스 정보를 포함하는 일반 경보의 예를 예시한 것이다.

[71] 도 22는 도 20의 일반 경보의 예에서 Type에 "-vm" 포스트픽스(postfix)를 붙인 예를 예시한 것이다.

[72] 도 23은 UI 경보를 예시한 것이다.

[73] 도 24는 VM을 타겟으로 하는 UI 경보를 구현하기 위한 실시예를 예시한 것이다.

[74] 도 25는 VM을 위한 사용자 상호 작용을 수행하는 절차를 예시한 것이다.

[75] 도 26은 UI 경보와 사용자 입력 정보를 예시한 것이다.

[76] 도 27은 본 발명이 적용될 수 있는 디바이스 및 서버를 예시한다.

발명의 실시를 위한 형태

[77] 본 발명은 가상 머신(Virtual Machine)에 관한 것이다. 하지만, 본 발명은 가상 머신에만 한정되는 것은 아니며, 본 발명의 기술적 사상이 적용될 수 있는 모든 시스템 및 방법에 적용될 수 있다.

[78] 본 명세서에서 사용되는 기술적 용어는 단지 특정한 실시예를 설명하기 위해 사용되는 것으로, 본 발명을 한정하려는 의도가 아님을 유의해야 한다. 또한, 본 명세서에서 사용되는 기술적 용어는 본 명세서에서 특별히 다른 의미로 정의되지 않는 한, 본 발명이 속하는 기술 분야에서 통상의 지식을 가진 자에 의해 일반적으로 이해되는 의미로 해석되어야 하며, 과도하게 포괄적인 의미로 해석되거나, 과도하게 축소된 의미로 해석되지 않아야 한다. 또한, 본 명세서에서 사용되는 기술적인 용어가 본 발명의 사상을 정확하게 표현하지 못하는 잘못된 기술적 용어일 때에는, 당업자가 올바르게 이해할 수 있는 기술적 용어로 대체되어 이해되어야 할 것이다. 또한, 본 발명에서 사용되는 일반적인 용어는 사전에 정의되어 있는 바에 따라, 또는 전후 문맥상에 따라 해석되어야 하며, 과도하게 축소된 의미로 해석되지 않아야 한다.

[79] 이하, 첨부된 도면을 참조하여 본 발명에 따른 바람직한 실시예를 상세히 설명한다. 도면 부호에 관계없이 동일하거나 유사한 구성 요소는 동일한 참조 번호를 부여하고 이에 대한 중복되는 설명은 생략하기로 한다. 또한, 첨부된 도면은 본 발명의 사상을 쉽게 이해할 수 있도록 하기 위한 것일 뿐 첨부된 도면에 의해 본 발명의 사상이 제한되는 것으로 해석되어서는 아니됨을 유의해야 한다. 본 발명의 사상은 첨부된 도면외에 모든 변경, 균등물 내지 대체물에 까지도 확장되는 것으로 해석되어야 한다.

[80] 이하 도면들에서 디바이스가 도시되어 있으나, 디바이스는 휴대폰, PDA, 스마트 폰(Smart Phone), 무선 모뎀(Wireless Modem), 노트북 등과 같이 통신 기능을 갖춘 휴대용 장치 뿐만 아니라 개인용 컴퓨터(Personal Computer, PC)나 차량 탑재 장치 등과 같이 휴대 불가능한 장치일 수 있다. 또한, 본 명세서에서

디바이스는 UE(User Equipment), ME(Mobile Equipment), MS(Mobile Station), UT(User Terminal), SS(Subscriber Station), 무선 디바이스(Wireless Device), 핸드헬드 디바이스(Handheld Device), AT(Access Terminal), 단말과 등가의 의미로 사용될 수 있다.

[81] 도 1은 하나의 디바이스 상에서 구성(configure)될 수 있는 가상 머신(VM)들을 예시한 것이다.

[82] 도 1을 참조하면, 예를 들어 엔터프라이즈 VM(Enterprise VM), 리얼타임 VM(RealTime VM), 게스트 VM(Guest VM)과 같은 복수의 VM들이 하나의 디바이스 상에 구성될 수 있다. 엔터프라이즈 VM은 향상된 보안을 제공하므로 데이터가 외부로 전달될 수 없고 VM을 제거(delete)할 때 관련 데이터들이 폐기(dispose)된다. 리얼타임 VM은 실시간 동작(real-time operation)을 위한 VM이며 CPU를 할당할 때 높은 우선 순위를 가진다. 게스트 VM은 제한된 저장 공간을 가지고 모바일 지불 수단(mobile payment)이나 전화(phone call)/SMS 등이 제한된다.

[83] 원격의 서버가 단말에 이러한 VM을 생성하기 위해서는 가상 머신 이미지(VM Image)가 필요하다. VM 이미지도 또한 원격의 서버가 단말에 전송해주어야 하는데, 전송되는 가상 머신 이미지의 타입에 따라 가상 머신을 생성하는 과정이 달라질 수 있다. 가상 머신 이미지에는 하나의 VM만을 생성할 수 있는 타입과 복수의 VM을 포함하며 이들 VM을 계속적으로 생성하는 데 사용될 수 있는 타입이 있다. 서버는 이 둘을 구분하여 가상 머신을 생성하여야 한다. VM웨어나 기타 종래 기술에 이러한 고려가 없었음은 물론이며, VM 생성 과정과 VM 생성을 위한 VM 이미지 전달 과정을 서버 중심으로 제어할 수 없었다.

[84] VM 관리 기술은 엔터프라이즈 VM에서 문제되는데 엔터프라이즈 VM에서는 회사의 데이터를 보호하기 위해 원격에서 가상 머신을 잠글 수 있다. 잠긴 가상 머신 (Locked VM)의 경우 사용자는 VM에 접근할 수 없게 된다. 원격 서버에 의한 잠금/잠금해제(Lock/Unlock) 동작은 VM 내부 상태(실행, 중지 등)에 영향을 받지 말아야 하며 독립적으로 이루어져야 한다. 종래의 기술에는 잠금/잠금해제 동작과 VM 내부 상태가 독립적으로 운영하는 방안이 고려되어 있지 않아서 잠긴 VM은 VM의 실행이 중지되었다. 이 경우 사용자가 VM에 접근하지 못하도록 우선적으로 잠금되기 때문에, VM의 바이러스 감염 여부, 보안 정책 설정 여부, 실행 환경의 무결성 검사 등을 수행하려고 하는 경우 문제가 된다. 즉 잠긴 VM은 실행이 중지되어 원격의 서버가 VM 내부의 프로세스에 연결할 수가 없다. 따라서, 위에서 열거한 검사를 수행하기 위해 VM을 통째로 서버로 전송한 후에 VM 검사를 서버에서 직접 수행할 수밖에 없었다.

[85] 가상 머신이 잠긴 경우, 사용자는 VM에 접근할 수가 없고 원격의 서버가 잠금을 해제해주어야 한다. 즉 VM에 잠금/잠금해제 동작을 수행할 수 있는 엔티티는 서버밖에 없다. 만약, VM이 잠긴 경우, 단말에 설정되어 있는 서버 키나 기타 네트워크 설정이 잘못 되어 원격의 서버가 단말에 접속할 수 없게

되면 잠긴 VM을 다시 잠금해제할 수가 없게 된다. 또, 잠긴 VM을 특정 사용자가 잠금해제하려고 하는 경우, 서버에게 웹 등의 인터페이스를 통해 잠금 해제를 요청해야 하고 서버는 사용자의 요청에 의해 잠긴 VM을 잠금해제하게 된다. 이러한 시나리오는 특정 사용자가 VM을 잠금해제하는 경우에 원격의 서버에 요청을 해야 하는 불편함이 있다. 원격 서버에 요청을 보내는 과정은 웹을 사용한다는 가정하에 웹 로그인, 잠금 해제하려는 VM의 선택, 선택된 VM의 잠금 해제 요청 등 절차가 복잡하다. 따라서 잠긴 VM을 안정하게 편리하게 특정 사용자가 잠금해제해야 하지만, 종래 기술에는 이러한 고려가 없었다.

[86] 가상 머신 장치 관리(Device Management for VM)는 가상 머신의 펌웨어 업데이트, 소프트웨어 설치/업데이트/제거 등의 관리, 원격 제어 등을 가리키는 용어이다. 이는 기존의 단말에 국한되어 있던 관리 기능이 가상 머신에 확대된 개념으로서, 단말의 장치 관리 기능과 유사하다. 참고로 이러한 VM 장치 관리는 앞서 설명한 VM 관리(VM Management)(예, VM 생성 또는 제거)와는 별개의 개념으로서, 생성된 가상 머신의 장치 관리에 국한되는 용어이다. 단말 관리에는 대표적으로 OMA(Open Mobile Alliance) DM(Device Management) 기술이 사용되었는데, 이는 VM 장치 관리에도 적용될 수 있다. 단말 관리에 적용되었던 OMA DM 기술을 가상 머신에도 적용할 경우, VM 장치 관리를 위한 별도의 기술 개발 과정 없이 기존의 OMA DM 기술을 활용할 수 있어 효율적인 VM 장치 관리가 가능하다. OMA DM을 VM 장치 관리에 적용하기 위해 가상 머신이 OMA DM 클라이언트를 포함할 수도 있다. 하지만, 가상 머신을 생성할 때마다 OMA DM 클라이언트를 생성해 주어야 하고 새로운 OMA DM 클라이언트에 새로운 디바이스 ID(Device Identifier)를 할당하고 부트스트랩(bootstrap)을 수행해 주어야 하는 문제 등이 발생한다. 따라서, 기존의 OMA DM 기술을 VM 장치 관리에 적용하기 위해서는 단말에 있는 단일 DM 클라이언트로 다수의 가상 머신을 장치 관리할 수 있는 기술이 필요하다.

[87] 도 2는 OMA DM 아키텍처를 기반으로 한 가상화 관리 오브젝트(Virtualizaiton Management Object, Virtualization MO, VirMO) 아키텍처를 예시한 것이다.

[88] 도 2를 참조하면, 5개의 엔티티(entity)가 도시되어 있는데 각 엔티티에 대해 설명하면 다음과 같다.

[89] ■ VirMO 클라이언트

[90] VirMO 클라이언트(예, VirMO 클라이언트(220))는 VirMO 서버(VirMO 서버(250))와 통신하며 VirMO 서버로부터 수신되는 VM 관리 동작(Virtual Machine management operation)을 지시(issue)하는 것을 담당하는 논리적 엔티티(logical entity)이다. VirMO 클라이언트는 응답으로서 이러한 동작의 결과를 VirMO 서버로 전송하고 VirMO 서버에게 일반 경보(Generic Alert)를 VirMO 서버로 보낸다. VirMO 클라이언트가 VM(예, VM(230))을 관리하는 데에는 디바이스 내부 인터페이스가 사용될 수 있다. VirMO 클라이언트(220)와 VirMO 서버(250)가 별도의 인터페이스를 가지는 것으로 도시되어 있지만,

실제로는 VirMO 클라이언트는 VirMO 서버와 통신하기 위해 DM 클라이언트(예, DM 클라이언트(210))를 이용할 수 있다.

[91] ■ VirMO 서버

[92] VirMO 서버(예, VirMO 서버(250))는 VirMO 클라이언트(VirMO 클라이언트(220))와 통신하며 VirMO 클라이언트에게 VM 관리 동작을 지시하는 것을 담당하는 논리적 엔티티이다. 또한 VirMO 서버는 VirMO 클라이언트로부터 일반 경보(Generic Alert)를 수신하고 소비(consume)한다. 마찬가지로, VirMO 클라이언트(220)와 VirMO 서버(250)가 별도의 인터페이스(예, VirMO-1 및 VirMO-2)를 가지는 것으로 도시되어 있지만, 실제로는 VirMO 서버는 VirMO 클라이언트와 통신하기 위해 DM 서버(예, DM 서버(240))를 이용할 수 있다.

[93] ■ DM 클라이언트

[94] VirMO 클라이언트(예, VirMO 클라이언트(220))와 VirMO 서버(예, VirMO 서버(250))는 DM 클라이언트(예, DM 클라이언트(210))와 DM 서버(예, DM 서버(240)) 간의 인터페이스인 DM 프로토콜(예, DM-1)을 통해 서로 통신할 수 있다. DM 클라이언트는 복수의 VM들을 관리하는 것을 담당하는 VirMO 클라이언트와 내부적으로 통신할 수 있다. DM 서버는 VirMO로 제공되는 디바이스 내의 관리 오브젝트를 DM 프로토콜을 통해 볼 수 있으며 관리 오브젝트를 조종할 수 있다. DM 클라이언트는 일반 경보(Generic Alert) 메커니즘을 이용하여 클라이언트에서 발생된 정보를 서버 측으로 전송할 수 있는데, VirMO 클라이언트 역시 이러한 메커니즘을 이용할 수 있다.

[95] ■ DM 서버

[96] VirMO 아키텍처는 DM 메시지 전송 및 디바이스 탐색(device discovery)을 위한 DM 서버 콤포넌트를 필요로 한다. DM 서버(예, DM 서버(240))는 일반 경보(Generic Alert) 메커니즘을 통해 VirMO 클라이언트(예, VirMO 클라이언트(220))로부터 수신되는 최종 통지(final notification)를 수신할 수 있다.

[97] ■ 가상 머신(VM)

[98] 가상머신(VM)(예, VM(230))은 독자적인 실행 환경으로서 VirMO 서버(예, VirMO 서버(250))에 의해 관리될 수 있다. VirMO 클라이언트는 VirMO 서버로부터 수신된 명령(instruction)에 따라 디바이스 내부 인터페이스를 통해 VM을 조종(manipulate)할 수 있다.

[99] 또한, 도 2를 참조하면, 3개의 인터페이스가 도시되어 있는데 이들 인터페이스에 대해 설명하면 다음과 같다.

[100] ■ VirMO-1

[101] VirMO-1 인터페이스는 VirMO 서버(예, VirMO 서버(250))가 VirMO 클라이언트(예, VirMO 클라이언트(220))에게 가상 머신 관리 동작을 수행하게끔 할 수 있다. 또한, VirMO 서버는 VirMO-1 인터페이스를 통해 VirMO 클라이언트로부터 상태 및 결과를 수신할 수 있다. VirMO-1 인터페이스는 아래

계층의 DM-1 인터페이스를 이용할 수 있다.

[102] ■ VirMO-2

[103] VirMO-2 인터페이스는 VirMO 클라이언트(에, VirMO 클라이언트(220))가 아래 계층의 DM-1 인터페이스를 이용하여 VirMO 서버(에, VirMO 서버(250))에게 일반 경보(Generic Alert)를 보낼 수 있게 한다.

[104] ■ DM-1

[105] DM-1 인터페이스는 OMA DM 인에이블러(Enabler)에 정의되어 있는 인터페이스이다. DM-1 인터페이스는 서버가 디바이스 관리 명령(device management command)을 클라이언트로 보내고 클라이언트가 상태 및 경보를 서버로 돌려 보낼 수 있는 인터페이스를 제공한다.

[106] 도 3은 본 발명에 적용될 수 있는 가상화 타입을 예시한 것이다. 타입 I과 타입 II의 2가지 가상화 타입의 예가 예시되어 있지만 가상화 방식은 이들 예로만 제한되는 것은 아니다.

[107] 도 3(a)는 타입 I 가상화를 예시한다. 도 3(a)를 참조하면, 타입 I 가상화의 경우 하이퍼바이저(에, 하이퍼바이저(320))가 직접적으로 디바이스 하드웨어(에, 하드웨어(310)) 상에서 동작하며 OS들을 위한 하드웨어 추상화 계층(abstract hardware layer)을 제공한다. 각 OS는 하나의 VM을 포함할 수 있으며 하이퍼바이저는 그 위에 존재하는 복수의 VM들(에, 관리 VM(332), 엔터프라이즈 VM(334), 개인용 VM(336))을 운영한다. 관리 VM은 다른 VM들을 관리하는 가상 머신 관리자(Virtual Machine Manager)로서 역할하며 다른 VM들을 생성하고 제거할 수 있다. 관리 VM은 전형적으로 심플 OS(Simple OS) 상에서 동작하며, 심플 OS는 이러한 목적들을 제공하기에 충분한 정도의 제한된 기능들만을 제공하는 최소 기능 세트(minimal set of functions)를 가질 수 있다.

[108] 도 3(b)는 타입 II 하이퍼바이저를 예시한다. 도 3(b)를 참조하면, 타입 II 가상화의 경우 하이퍼바이저(에, 하이퍼바이저(340))는 전형적인 OS(에, OS(350)) 내에서 동작하며 OS 내부에서 생성된 VM들(에, 엔터프라이즈 VM(342), 개인용 VM(344))을 위한 하드웨어 추상화 계층을 제공한다.

[109] 도 3의 예들에서 가상화가 하이퍼바이저를 통해 이루어질 수 있으므로 가상화 타입은 하이퍼바이저 타입으로 지칭될 수 있다. 따라서, 타입 I 가상화는 타입 I 하이퍼바이저로 지칭될 수 있고, 타입 II 가상화는 타입 II 하이퍼바이저로 지칭될 수 있다.

[110] 도 4는 타입 I 가상화를 통해 가상화 관리 오브젝트를 위한 엔티티들이 구현되는 예를 예시한다.

[111] 도 4를 참조하면, DM 클라이언트(에, DM 클라이언트(210))와 VirMO 클라이언트(에, VirMO 클라이언트(220))는 관리 VM(에, 관리 VM(332))내에 구현될 수 있다. VirMO 클라이언트는 디바이스 내의 내부 인터페이스를 통해 각 VM(에, 엔터프라이즈 VM(334), 개인용 VM(336))과 통신할 수 있다. 또한, VirMO 클라이언트는 하이퍼바이저(에, 하이퍼바이저(320))와 통신할 수 있다.

VirMO 클라이언트는 VirMO 서버(예, VirMO 서버(250))로부터 수신되는 관리 동작들을 수행함으로써 각 VM을 관리할 수 있다.

[112] 도 5는 타입 II 가상화를 통해 가상화 관리 오브젝트를 위한 엔티티들이 구현되는 예를 예시한다.

[113] 도 5를 참조하면, VirMO 클라이언트(예, VirMO 클라이언트(220))와 DM 클라이언트(예, DM 클라이언트(210))는 OS(예, OS(350)) 내에 구현될 수 있다. VirMO 클라이언트는 하이퍼바이저(예, 하이퍼바이저(340)) 및 각 VM(예, 엔터프라이즈 VM(334), 개인용 VM(336))과 통신할 수 있으며, VirMO 서버(예, VirMO 서버(250))로부터 수신되는 관리 동작(management operation)들을 수행할 수 있다.

[114] 도 4와 도 5를 참조하여 설명한 바와 같이, VirMO 서버는 디바이스 내의 VM을 관리할 수 있다. 이러한 관리 동작은 VM을 생성(create), 제거(delete), 정지(suspend)시키는 것을 포함할 수 있다. 이러한 관리 동작들은 VirMO 클라이언트에 의해 수행될 수 있으며, 수행 결과로서 VM의 상태가 변경될 수 있다. VM의 상태(state)는 VM이 현재 처한 조건(condition)을 나타낸다.

[115] 도 6은 VM의 가능한 상태와 프리머티브(primitive)를 예시하는 상태 머신(state machine)의 예를 도시한 것이다.

[116] 도 6을 참조하면, 예를 들어, VM의 가능한 상태들은 생성전 상태(NotCreated state)(610), 생성 상태(Created state)(620), 실행 상태(Running state)(630), 정지 상태(Suspended state)(640), 파워오프 상태 (PoweredOff state)(650), 제거 상태(Deleted state)(660)를 포함할 수 있다. 또한, VM 관리 프리머티브(VM management primitive)는 이러한 상태들 간의 천이(transition)를 트리거링(trigger)한다. 예를 들어, VM 관리 프리머티브는 생성(Create), 정지(Suspend), 재개(Resume), 제거(Delete), 파워오프(Poweroff), 파워온(Poweron), 생성 및 파워오프(CreateAndPoweroff) 프리머티브를 포함할 수 있다.

[117] 디바이스 내부 동작들과 상태 천이는 원자성(atomic)일 수 있다. 원자성 동작(atomic operation)은 하나의 동작이 여러 개의 다른 동작들로 나뉘어질 수 없는 동작을 의미한다. 따라서, 상태 천이가 실패하는 경우, VirMO 클라이언트는 동작을 되돌릴 수 있으며 VM은 동작을 실행하기 이전 상태에 머물러 있을 수 있다.

[118] 도 6에 예시된 각 상태를 설명하면 다음과 같다.

[119] ■ 생성전 상태(NotCreated state)

[120] 생성전 상태는 VM가 디바이스 내에 생성되지 않은 상태이다. 생성전 상태에서 VM을 생성하는 데 사용되는 정보가 미리 설정(pre-configure)될 수 있다.

[121] ■ 생성 상태(Created state)

[122] 생성 상태는 실행 상태, 정지 상태, 파워오프 상태를 포함할 수 있다. 일 예로, 생성 상태는 실행 상태, 정지 상태, 파워오프 상태가 총괄적으로(collectively)

그룹화된 상태일 수 있다. 생성 상태에서 VM은 생성되었으며, 이는 VM을 위한 가상 실행 환경(virtual execution environment)이 실행(running)을 위해 셋업되고 준비된 것을 의미할 수 있다.

[123] ■ 실행 상태(Running state)

[124] 실행 상태는 VM이 생성되었으며 VM을 위한 가상 실행 환경이 셋업되고 실행중인 것을 의미할 수 있다. 실행 상태에서 운영 체제, 애플리케이션 및 다른 콤포넌트들이 VM 상에 설치(install)되고 실행(execute)될 수 있다. VM은 이들이 적절하게 설치되고 실행될 수 있도록 요구되는 자원들을 할당할 수 있다

[125] ■ 정지 상태(Suspended state)

[126] 정지 상태는 VM이 생성되었지만 VM의 실행이 중지(stop)된 것을 의미할 수 있다. 정지 상태에서 VM의 모든 애플리케이션들과 운영 체제는 실행되지 않을 수 있다. 또한 정지 상태에서 VM의 현재 실행 상태(execution state)는 저장도리 수 있다. 일단 정지되면, VM을 위해 할당된 자원들은 다른 데 활용될 수 있도록 해제(release)될 수 있다. 하지만, 이들 할당된 자원들은 예를 들어 VM이 정지 상태로부터 실행 상태로 천이되어 재개(resume)되면 재할당될 수 있다.

[127] ■ 파워오프 상태(PoweredOff state)

[128] 파워오프 상태는 VM이 생성되었지만 파워 오프된 것을 의미할 수 있다. VM이 파워 오프되면, VM 상에서 실행중인 운영 체제 또는 애플리케이션들은 함께 종료할 수 있다. VM을 재실행중으로 만들기 위해서는 정상적인 운영 체제 부팅 프로세스(boot process)를 통해 VM을 파워 온하는 것을 필요로 한다.

[129] ■ 제거 상태(Deleted state)

[130] 제거 상태는 VM이 디바이스로부터 완전히 제거되었으며 VM에 할당된 자원들이 해제되고 액세스될 수 없는 것을 의미할 수 있다. VM에 속한 데이터는 파기(destroy)될 수 있다. 예를 들어, 제거 상태에서 VirMO 클라이언트는 디바이스로부터 VM을 제거하고 관리 트리(management tree)로부터 VM을 위한 노드(node)들을 제거할 수 있다.

[131] 표 1은 도 6에 예시된 프리머티브들에 대한 설명과 이들 프리머티브들이 적용될 수 있는 상태(applicable state) 및 적용후 상태(post-primitive state)를 예시한 것이다.

[132] 표 1

[Table 1]

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

[133] 표 1을 참조하여 각 프리머티브를 설명하면 다음과 같다.

[134] ■ 생성(Create) 프리머티브

[135] 생성(Create) 프리머티브는 VM을 생성한다. 생성 프리머티브는 생성전 상태(610)에서 적용될 수 있다. 생성 프리머티브에 의해 VM은 실행 상태(630)로 천이할 수 있다.

[136] ■ 생성 및 파워오프(CreateAndPoweroff) 프리머티브

[137] 생성 및 파워오프(CreateAndPoweroff) 프리머티브는 VM을 생성하지만 생성된 VM을 파워오프 상태(650)에 있게 한다. 생성 및 파워오프 프리머티브는 생성전 상태(610)에서 적용될 수 있다. 생성 및 파워오프 프리머티브에 의해 VM은 파워오프 상태(650)로 천이할 수 있다.

[138] ■ 재개(Resume) 프리머티브

[139] 재개(Resume) 프리머티브는 VM을 재개하여 VM을 실행 중이도록 만든다. 재개 프리머티브는 정지 상태(640)에서 적용될 수 있고, 재개 프리머티브에 의해 VM은 실행 상태(630)로 천이할 수 있다.

[140] ■ 정지(Suspend) 프리머티브

[141] 정지(Suspend) 프리머티브는 VM을 정지시킨다. 정지 프리머티브는 실행 상태(630)에서 적용될 수 있고, 정지 프리머티브에 의해 VM은 정지 상태(640)로 천이할 수 있다.

[142] ■ 파워온(Poweron) 프리머티브

[143] 파워온(Poweron) 프리머티브는 VM을 파워 온시킨다. 파워온 프리머티브는 파워오프 상태(650)에서 적용될 수 있고, 파워온 프리머티브에 의해 VM은 실행 상태(630)로 천이할 수 있다.

[144] ■ 파워오프(Poweroff) 프리머티브

[145] 파워오프(Poweroff) 프리머티브는 VM을 파워 오프시킨다. 파워오프 프리머티브는 실행 상태(630)에서 적용될 수 있고, 파워오프 프리머티브에 의해 VM은 파워오프 상태(650)로 천이할 수 있다.

[146] ■ 제거(Delete) 프리머티브

[147] 제거(Delete) 프리머티브는 디바이스에서 VM을 제거한다. 제거 프리머티브는

생성 상태(620)에서 적용될 수 있고, 제거 프리머티브에 의해 VM은 제거 상태(660)로 천이할 수 있다.

[148] 본 발명에 따른 상태 및 프리머티브들은 도 6에 예시된 상태 및 프리머티브들로만 제한되는 것은 아니며 다른 상태 및 프리머티브들을 포함할 수 있다. 예를 들어, 본 발명에 따른 상태 머신은 제거중 상태(Deleting state), 포어그라운드 상태(Foreground state), 백그라운드 상태(Background state), 잠금 상태(Locked state), 잠금해제 상태(Unlocked state)를 포함할 수 있다. 또한, 본 발명에 따른 상태 머신은 프론트(Front), 프론트(Front), 백(Back), 잠금(Lock), 잠금해제(Unlock), 생성 및 정지(CreateSuspend) 프리머티브들을 포함할 수 있다.

[149] 도 7은 추가적인 VM의 가능한 상태와 프리머티브를 포함하는 상태 머신의 예를 예시한 것이다.

[150] 도 7을 참조하면, 생성 상태(Created state)(620)에서 VM은 포어그라운드 상태(710)와 백그라운드 상태(720)를 포함할 수 있다. 이들 상태들에 대해 설명하면 다음과 같다.

[151] ■ 포어그라운드 상태(Foreground state)

[152] 포어그라운드 상태(Foreground state)는 VM이 사용자와 직접적으로 정보를 교환하고 상호 작용할 수 있는 상태를 말한다. 포어그라운드 상태에서, VM은 입력 포커스(input focus)를 가지며 입출력(Input/Output, IO) 디바이스와 사용자 인터페이스를 이용하여 사용자와 상호 작용을 수행할 수 있다. 예를 들어, 사용자는 키보드, 마우스, 마이크, 터치 스크린 등과 같은 UI(User Interface) 디바이스를 이용하여 VM과 상호 작용 할 수 있다. VM이 사용자와 상호 작용을 하기 위해서는 포어그라운드 상태에 있어야 하며, 특정 시점에 사용자와 상호작용 가능한 VM은 일반적으로 하나이기 때문에 포어그라운드 상태에 있는 VM은 최대 1개라고 볼 수 있다.

[153] ■ 백그라운드 상태(Background state)

[154] 백그라운드 상태(Background state)는 VM이 사용자와 직접적인 상호 작용이 가능하지 않는 상태를 말한다. 백그라운드 상태에서, VM은 입력 포커스를 가지지 않고 사용자와 상호 작용을 수행할 수 없다. 포어그라운드 상태와 백그라운드 상태는 단말에 따라 지원되지 않을 수도 있다.

[155] 포어그라운드 상태 및 백그라운드 상태와는 별도로/부가적으로 생성 상태(Created state)(620)에서 VM은 잠금 상태(Locked state)(730)와 잠금해제 상태(Unlocked state)(740)를 포함할 수 있다. 이들 상태들에 대해 설명하면 다음과 같다.

[156] ■ 잠금 상태(Locked state)

[157] 잠금 상태(Locked state)는 VM이 잠겨 있는 상태이다. 잠금 상태에서 VM에 대한 권한 없는 액세스(unauthorized access)는 불가능할 수 있다. 즉, 잠금 상태에서 VM은 인증 정보 등을 통해 권한을 가지는 엔티티(또는 사용자)만 액세스할 수 있다. 예를 들어, 잠금 상태는 단말에서 초기 화면이 잠겨있는

상태일 수 있다.

[158] ■ 잠금해제 상태(Unlocked state)

[159] 잠금해제 상태(Unlocked state)는 잠금 상태에 반대되는 개념으로 VM에 대한 액세스가 제한되지 않는 상태이다. 잠금해제 상태에서 VM은 잠금된 기능들이 다시 사용 가능할 수 있다. 잠금 상태와 잠금해제 상태는 단말에 따라 지원되지 않을 수 있다.

[160] 또한, 도 7에 도시되지 않았지만 VM은 제거중 상태(Deleting state)를 포함할 수 있다. 제거중 상태는 VM이 제거 프리머티브에 의해 제거가 진행되고 있음을 나타낸다. VM의 제거는 시간이 상당히 걸릴 수가 있고 제거 과정에 있는 VM이 여전히 실행 상태나 정지 상태에 있게 되어, VirMO Server가 VM의 정확한 상태를 알 수가 없다. 따라서, 제거 과정에 있는 VM은 관리 동작(management operation)을 수행하는 데 제약이 있을 수 있다.

[161] ■ 제거중 상태(Deleting state)

[162] 제거중 상태(Deleting state)는 VirMO 클라이언트가 VM 제거를 시작하였지만 VM이 완전히 제거되지 않은 상태이다. 제거(Delete) 프리머티브가 실행되면, VM은 제거중 상태로 우선 천이하게 되고, VM을 실제 단말에서 제거하는 과정이 진행된다. VM 제거가 완료되면, VM은 제거 상태(Deleted state)로 천이한다. 제거중 상태에서 제거 상태로 천이하기 위해 별도의 프리머티브를 필요로 하지 않으며, VM 제거 완료 여부에 따라 단말이 자동으로 업데이트 하게 된다. 제거중 상태에서 VirMO 클라이언트는 VM을 위한 관리 동작을 제대로 처리할 수 없다.

[163] 표 2는 위에서 기술된 추가적인 프리머티브들에 대한 설명과 이들 프리머티브들이 적용될 수 있는 상태(applicable state) 및 적용후 상태(post-primitive state)를 예시한 것이다.

[164] 표 2

[Table 2]

Primitives	Descriptions	Applicable State	Post-Primitive State
CreateSuspend	To create the VM. The created VM is in the Suspended state	NotCreated	Suspended
Front	To transit the VM to the Foreground state	Created:Background	Created:Foreground
Back	To transit the VM to the Background state	Created:Foreground	Created:Background
Lock	To lock the VM	Created:Unlocked	Created:Locked
Unlock	To unlock the VM	Created:Locked	Created:Unlocked

[165] 표 2를 참조하여 각 프리머티브를 설명하면 다음과 같다.

[166] ■ 생성 및 정지(CreateSuspend) 프리머티브

[167] 생성 및 정지(CreateSuspend) 프리머티브는 VM을 생성하지만 생성된 VM을 정지 상태(640)에 있게 한다. 생성 및 정지 프리머티브는 생성전 상태(610)에서

적용될 수 있고, 생성 프리머티브에 의해 VM은 정지 상태(640)로 천이할 수 있다.

[168] ■ 프론트(Front) 프리머티브

[169] 프론트(Front) 프리머티브는 생성된 VM을 포어그라운드 상태(710)에 있게 한다. 프론트 프리머티브는 백그라운드 상태(720)에서 적용될 수 있고, 프론트 프리머티브에 의해 VM은 포어그라운드 상태(710)로 천이할 수 있다.

[170] ■ 백(Back) 프리머티브

[171] 백(Back) 프리머티브는 생성된 VM을 백그라운드 상태(720)에 있게 한다. 백 프리머티브는 포어그라운드 상태(710)에서 적용될 수 있고, 백 프리머티브에 의해 VM은 백그라운드 상태(720)로 천이할 수 있다.

[172] ■ 잠금(Lock) 프리머티브

[173] 잠금(Lock) 프리머티브는 VM을 잠근다. 잠금 프리머티브는 잠금해제 상태(740)에서 적용될 수 있고, 잠금 프리머티브에 의해 VM은 잠금 상태(730)로 천이할 수 있다.

[174] ■ 잠금해제(Unlock) 프리머티브

[175] 잠금해제(Unlock) 프리머티브는 VM을 잠금해제한다. 잠금해제 프리머티브는 잠금 상태(730)에서 적용될 수 있고, 잠금해제 프리머티브에 의해 VM은 잠금해제 상태(740)로 천이할 수 있다.

[176] 이상에서 본 발명이 적용될 수 있는 상태들과 관련 프리머티브들을 예시하였지만 본 발명은 이들 상태 및 프리머티브들로만 제한되는 것은 아니다. 또한, 실시예에 따라 상기 기술된 상태들과 프리머티브들 중 일부는 생략되고 실시될 수 있다. 또한, 실시예들에 따라 각 상태들과 프리머티브들은 필요에 따라 조합하여 실시될 수 있다. 일 예로, 생성 상태는 정지 상태와 실행 상태만 포함하고 파워오프 상태는 제외될 수 있다. 따라서, 생성 및 파워오프 프리머티브는 제외될 수 있다. 다른 예로, 생성 상태에서 파워오프 상태가 제외될 수 있으며 생성 및 파워오프 프리머티브 대신 생성 및 정지 프리머티브가 포함될 수 있다.

[177] 도 8은 VM 관리를 위한 가상화 관리 오브젝트의 예를 예시한 것이다. 가상화 관리 오브젝트(Virtualizaiton Management Object, VirMO)는 VirMO 서버와 VirMO 클라이언트 간에 가상 머신(VM)을 관리하는 데 사용되는 인터페이스를 제공한다. 가상화 관리 오브젝트는 단말의 장치 관리 트리(즉, 단말 자체의 장치 관리 트리)에 포함될 수 있다.

[178] 도 8을 참조하면, 가상화 관리 오브젝트(VirMO)는 복수의 노드들을 포함할 수 있는데 각 노드에 대해 설명하면 다음과 같다. 다음 설명에서 트리 발생 횟수(Tree Occurence)는 해당 트리 구조에서 각 노드의 발생 횟수를 나타내며, "One"은 1번 발생하는 것을 나타내고, "ZeroOrMore"는 0번 이상 발생하는 것을 나타내고, "ZeroOrOne"는 0번 또는 1번 발생하는 것을 나타낸다. 또한, 포맷(Format)은 내용 정보의 포맷을 나타내며, "int"는 정수 포맷을 나타내고,

"bin"는 이진 포맷을 나타내고, "chr"는 캐릭터 포맷을 나타내고, "node"는 트리의 내부 오브젝트(interior object)임을 나타내고, "null"은 내용 정보가 없음을 나타낸다. 최소 액세스 타입(Min. Access Type)은 각 노드에 대해 최소한 액세스 권한이 부여되는 동작 타입을 나타내며, "Get"은 가져오기 명령(예, Get command)이 최소한 허용됨을 나타내고, "No Replace"는 치환 명령(예, Replace command)이 허용되지 않음을 나타내고, "Exec"는 실행 명령(예, Exec command)이 최소한 허용됨을 나타낸다.

[179] 또한, 본 명세서에서 노드는 OMA DM 프로토콜을 통해 수행되는 관리 동작들에 의해 관리될 수 있는 개체를 의미한다. 관리 트리 구조를 구성하는 한 구성 요소를 노드라고 지칭할 수 있다. 예를 들어, 관리 트리에서 루트 노드는 디바이스를 나타낼 수 있다. 또한, 관리 트리에서 노드는 디바이스 내에 포함되는 엔티티들을 나타낼 수 있다. 일 예로, 노드는 가상 머신을 나타낼 수 있다. 이외에도, 노드는 다양한 정보를 저장하거나 자식 노드를 가질 수 있다.

[180] ■ VirMO

[181]

Tree Occurrence	Format	Min. Access Types
One	node	Get

[182] VirMO 노드는 이 가상화 MO에 대한 루트 노드(root node)이다. DM 트리(tree)에서 가상화 MO의 위치는 이 노드의 조상 엘리먼트(ancestor element)에 의해 정해진다. 가상화 MO에 대한 식별자(예, MOID)는 예를 들어 "urn:oma:mo:oma-VirMO:1.0"일 수 있다.

[183] ■ VirMO/<x>

[184]

Tree Occurrence	Format	Min. Access Types
ZeroOrMore	node	Get

[185] VirMO/<x> 노드는 특정 가상 머신에 대한 파라미터들을 위한 플레이스홀더(placeholder)이다. 관리되는 가상 머신들 각각에 대한 해당 파라미터들은 이 노드 아래에서 이용 가능하다.

[186] ■ VirMO/<x>/VMID

[187]

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

[188] VirMO/<x>/VMID 노드는 캐릭터(character) 값을 가지며 가상 머신에 대한 식별자를 특정한다. 이 노드에 대한 값은 VirMO 클라이언트에 이해 할당될 수 있으며 그 값은 가상화 MO 내에서 고유(unique)하다.

[189] ■ VirMO/<x>/State

[190]

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

[191] VirMO/<x>/State 노드는 정수(integer) 값을 가지며 가상 머신의 현재 상태를 특정한다. 이 노드의 값은 VirMO 클라이언트에 의해 설정될 수 있다. 그 값은 다음과 같다.

[192]

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

[193] 예시된 상태 값들은 예시일 뿐이며, 실시예에 따라 그 값들은 달라질 수 있다. 일 예로, 실행 상태는 1의 값을 가지고, 정지 상태는 2의 값을 가지고, 제거 상태는 3의 값을 가질 수 있다. 다른 예로, 실행 상태는 1의 값을 가지고, 정지 상태는 2의 값을 가지고, 제거중 상태는 3의 값을 가지고, 제거 상태는 4의 값을 가질 수 있다. 이 외에도 다른 상태 및 값들의 조합이 가능하다.

[194] ■ VirMO/<x>/Operations

[195]

Tree Occurrence	Format	Min. Access Types
One	node	Get

[196] VirMO/<x>/Operations 노드는 가상화 MO에 의해 지원되는 동작들에 대한 부모 노드(parent node)이다. 이 노드는 VM 관리에 사용되는 동작들의 세트를 제공하는 서브-트리(sub-tree)이다.

[197] ■ VirMO/<x>/Operations/Create

[198]

Tree Occurrence	Format	Min. Access Types
One	null	Exec

[199] VirMO/<x>/Operations/Create 노드는 VirMO 서버가 가상 머신을 생성하기 위한 실행 명령(예, Exec command)으로 사용될 수 있다. 생성된 가상 머신은 실행 상태(Running state)에 있을 수 있다.

[200] ■ VirMO/<x>/Operations/CreateParams

[201]

Tree Occurrence	Format	Min. Access Types
One	chr	Get

[202] VirMO/<x>/Operations/CreateParams 노드는 VM을 생성하는 데 사용되는 파라미터들을 명시한다. VirMO 서버는 Create 노드를 실행하기 전에 VirMO/<x>/Operations/CreateParams 노드에 설정된 파라미터들을 설정할 수 있다. CreateParams 노드가 설정된 후에 VirMO 서버는 Create 노드를 실행할 수

있으며 디바이스는 CreateParams 노드에 따라 VM을 생성할 수 있다. 파라미터들은 예를 들어 "key=value" 쌍들의 시퀀스로서 구성될 수 있으며, 이 쌍들은 앰퍼샌드(ampersand) "&"에 의해 구분될 수 있다. 예시적인 파라미터들이 아래에 예시되어 있지만, 단말 제조자(vendor) 특정 확장 파라미터들도 가능하다. VirMO 서버와 VirMO 클라이언트는 아래의 예시적인 파라미터들을 실시예에 따라 일부 또는 전부 지원할 수 있다.

[203]

Parameters	Descriptions
DuplicateFrom=<VMID>	The created VM should be duplicated from the specified VM by the <VMID>.
ImgID=<ImgID>	The VirMO Client uses the image specified by the <ImgID> to create the VM. The <ImgID> is one of the ImgIDs in the VM Image MO.
DeleteSubTree=<bool>	If <bool> is true, the VM image sub-tree under the VMImgMO node used for creating the VM is deleted. If <bool> is false, the sub-tree for the VM image will remain, and can be used for creating other VMs over and over again. If not specified, the default value is false.
Running=<bool>	If <bool> is true, the created VM will be in the Running state. If <bool> is false, the created VM will be in the PowerOff state. If not specified, the default value is true.

[204] DuplicateFrom 파라미터는 <VMID> 값에 의해 특정된 VM을 복사(duplicate)하여 VM 생성할 때 사용될 수 있다. 이 때 생성된 VM의 VirMO/<x> 서브-트리는 <VMID> 값에 의해 특정된 원래의 VM과 동일한 구성을 가질 수 있다. 하지만, 이 경우에도 VMID 노드는 원래의 VM과 서로 달라야 하며, State 노드 역시 서로 다를 수 있다.

[205] ImgID 파라미터는 VM을 생성하는 데 사용되는 VM 이미지를 특정한다. <img_id> 값은 VMImgMO/<x>/ImgID 노드의 값들 중 하나일 수 있다.

[206] DeleteSubTree 파라미터는 불린(boolean) 값을 가진다. 그 값이 참(true)이면, VM을 생성하는 데 사용된 VMImgMO 노드 아래의 VM 이미지 서브-트리가 제거된다. 그 값이 거짓(false)이면, VM 이미지를 위한 서브-트리는 남아 있고 다른 VM들을 생성하는 데 반복적으로 사용될 수 있다. 디폴트 값은 거짓일 수 있다.

[207] Running 파라미터는 불린(boolean) 값을 가진다. 그 값이 참(true)이면, 생성된 VM은 실행(Running) 상태에 있을 수 있다. 그 값이 거짓(false)이면, 생성된 VM은 파워오프(PoweredOff) 상태에 있을 수 있다. 디폴트 값은 참일 수 있다.

[208] ■ VirMO/<x>/Operations/CreateSuspend

[209]

Tree Occurrence	Format	Min. Access Types
One	null	Exec

[210] VirMO/<x>/Operations/CreateSuspend 노드는 VirMO 서버가 가상 머신을 생성하기 위한 실행 명령(예, Exec command)으로 사용될 수 있다. 하지만, Create 노드와 달리 생성된 가상 머신은 정지 상태(Suspended state)에 있다.

[211] ■ VirMO/<x>/Operations/Suspend

[212]

Tree Occurrence	Format	Min. Access Types
One	null	Exec

[213] VirMO/<x>/Operations/Suspend 노드는 VirMO 서버가 가상 머신을 정지시키기 위한 실행 명령(예, Exec command)으로 사용될 수 있다. 실행이 성공적이면 VM의 상태는 정지 상태(Suspended state)에 있게 된다.

[214] ■ VirMO/<x>/Operations/Resume

[215]

Tree Occurrence	Format	Min. Access Types
One	null	Exec

[216] VirMO/<x>/Operations/Resume 노드는 VirMO 서버가 가상 머신을 재개시키기 위한 실행 명령(예, Exec command)으로 사용될 수 있다. 이 동작에 의해 VM의 상태는 정지 상태에서 실행 상태(Running state)로 천이된다.

[217] ■ VirMO/<x>/Operations/Delete

[218]

Tree Occurrence	Format	Min. Access Types
One	null	Exec

[219] VirMO/<x>/Operations/Delete 노드는 VirMO 서버가 가상 머신을 제거하기 위한 실행 명령(예, Exec command)으로 사용될 수 있다. 생성 상태(Created state)에 있는 가상 머신이 제거될 수 있다.

[220] ■ VirMO/<x>/Operations/Front

[221]

Tree Occurrence	Format	Min. Access Types
One	null	Exec

[222] VirMO/<x>/Operations/Front 노드는 VirMO 서버가 가상 머신을 포어그라운드 상태로 천이시키기 위한 실행 명령(예, Exec command)으로 사용될 수 있다. 생성된 상태(즉, 실행 상태, 정지 상태, 또는 파워오프 상태)에 있는 VM만이 대상이 될 수 있다. 이 명령에 의해 VM은 백그라운드 상태에서 포어그라운드 상태로 변경될 수 있다. 동시에 하나의 VM만이 포어그라운드 상태에 있을 수 있는 경우에는, 포어그라운드 상태에 있는 다른 VM은 자동으로 백그라운드 상태로 천이될 수 있다.

[223] ■ VirMO/<x>/Operations/Back

[224]

Tree Occurrence	Format	Min. Access Types
One	null	Exec

[225] VirMO/<x>/Operations/Back 노드는 VirMO 서버가 가상 머신을 백그라운드 상태로 천이시키기 위한 실행 명령(예, Exec command)으로 사용될 수 있다. 생성된 상태(즉, 실행 상태, 정지 상태, 또는 파워오프 상태)에 있는 VM만이 대상이 될 수 있다. 이 명령에 의해 VM은 포어그라운드 상태에서 백그라운드 상태로 변경될 수 있다.

[226] ■ VirMO/<x>/Operations/Lock

[227]

Tree Occurrence	Format	Min. Access Types
One	null	Exec

[228] VirMO/<x>/Operations/Lock 노드는 VirMO 서버가 가상 머신을 잠금하기 위한 실행 명령(예, Exec command)으로 사용될 수 있다.

[229] ■ VirMO/<x>/Operations/Unlock

[230]

Tree Occurrence	Format	Min. Access Types
One	null	Exec

[231] VirMO/<x>/Operations/Unlock 노드는 VirMO 서버가 가상 머신을 잠금해제하기 위한 실행 명령(예, Exec command)으로 사용될 수 있다.

[232] ■ VirMO/<x>/LockParams

[233]

Tree Occurrence	Format	Min. Access Types
One	node	Get

[234] VirMO/<x>/LockParams 노드는 잠금 동작(lock operation)과 관련된 파라미터들에 대한 부모 노드(parent node)이다.

[235] ■ VirMO/<x>/LockParams/State

[236]

Tree Occurrence	Format	Min. Access Types
One	int	Get

[237] VirMO/<x>/LockParams/State 노드는 가상 머신의 현재 잠금 상태를 특정한다. 이 값은 VirMO 클라이언트에 의해 설정될 수 있으며, 다음과 같은 값들을 가질 수 있다.

[238]

Valid Value	Descriptions
0	The VM is unlocked.
1	The VM is locked.

[239] 예를 들어, VirMO/<x>/LockParams/State 노드가 0의 값을 가지는 경우 VM은 잠금해제된 상태임을 나타낼 수 있다. VirMO/<x>/LockParams/State 노드가 1의

값을 가지는 경우 잠긴 상태임을 나타낼 수 있다. 노드의 값은 이 값들로만 제한되는 것은 아니며 실시예에 따라 변경될 수 있다.

[240] ■ VirMO/<x>/LockParams/UserUnlock

[241]

Tree Occurrence	Format	Min. Access Types
One	bool	Get

[242] VirMO/<x>/LockParams/UserUnlock 노드의 값은 사용자가 잠긴 가상 머신을 잠금해제하는 것이 허용되는지 여부를 특정한다. 이 노드는 불린 값을 가질 수 있다. 예를 들어, 노드 값이 참으로 설정된 경우 VirMO/<x>/VMID 노드에 의해 식별되는 가상 머신이 잠금 상태에 있더라도 사용자는 잠금해제하는 것이 허용된다. 노드 값이 거짓으로 설정된 경우 사용자는 잠긴 가상 머신을 잠금해제하는 것이 허용되지 않는다. 하지만, 이 노드의 값에 관계없이 권한을 가진(authorized) VirMO 서버는 가상머신을 잠금해제할 수 있다.

[243] ■ VirMO/<x>/LockParams/UnlockAuth

[244]

Tree Occurrence	Format	Min. Access Types
ZeroOrOne	node	Get

[245] VirMO/<x>/LockParams/UnlockAuth 노드는 잠금해제 인가 정보(unlock authorization information)에 관한 파라미터들에 대한 부모 노드이다. VirMO 클라이언트는 사용자가 VM을 잠금해제할 때 이 정보를 이용하여 사용자를 인가(authorize)할 수 있다. 이 노드는 UserUnlock 노드의 값이 참으로 설정된 경우 존재할 수 있다.

[246] ■ VirMO/<x>/LockParams/UnlockAuth/AuthType

[247]

Tree Occurrence	Format	Min. Access Types
One	int	Get

[248] VirMO/<x>/LockParams/UnlockAuth/AuthType 노드는 가상 머신을 잠금해제하는 데 대한 인가 타입(authorization type)을 특정할 수 있다. 이 노드는 예를 들어 다음과 같은 정수 값을 가질 수 있다.

[249]

Values	Description
0	DIGEST-SHA256 as specified in RFC6234

[250] 일 예로, VirMO/<x>/LockParams/UnlockAuth/AuthType 노드가 0의 값을 가지는 경우 RFC6234에 명시된 DIGEST-SHA256 타입을 나타낼 수 있다.

[251] ■ VirMO/<x>/LockParams/UnlockAuth/AuthData

[252]

Tree Occurrence	Format	Min. Access Types
One	bin	Get

[253] VirMO/<x>/LockParams/UnlockAuth/AuthData 노드는 AuthType 노드에 의해

특정된 인가 타입에 관련된 인가 데이터(authorization data)를 특정할 수 있다.

[254] ■ VirMO/<x>/LockParams/UnlockAuth/AuthDigest

[255]

Tree Occurrence	Format	Min. Access Types
One	bin	Get

[256] VirMO/<x>/LockParams/UnlockAuth/AuthDigest 노드는 AuthType 노드에 의해 특정된 인가 타입에 관련된 인가 다이제스트(authorization digest)를 특정할 수 있다.

[257] 이상에서, 도 8을 참조하여 가상화 MO가 가질 수 있는 여러 노드들을 설명하였다. 하지만, 이러한 노드들은 오로지 예시일 뿐이며 실시예에 따라 다른 노드들을 추가로 포함할 수 있다. 또한, 실시예에 따라 상기 설명된 노드들 중 일부는 제외될 수 있다. 또한, 실시예에 따라 상기 노드들의 위치도 변경될 수 있다. 또한, 실시예에 따라 노드들의 이름도 변경될 수 있다.

[258] 예를 들어, 가상화 MO는 다음과 같은 노드들을 추가로 포함할 수 있다.

[259] ■ VirMO/<x>/Operations/PowerOn

[260]

Tree Occurrence	Format	Min. Access Types
One	null	Exec

[261] VirMO/<x>/Operations/PowerOn 노드는 VirMO 서버가 가상 머신을 파워 온하기 위한 실행 명령(예, Exec command)으로 사용될 수 있다.

[262] ■ VirMO/<x>/Operations/PowerOff

[263]

Tree Occurrence	Format	Min. Access Types
One	null	Exec

[264] VirMO/<x>/Operations/PowerOff 노드는 VirMO 서버가 가상 머신을 파워 오프하기 위한 실행 명령(예, Exec command)으로 사용될 수 있다.

[265] 다른 예로, 잠금(Lock) 및 잠금해제(Unlock) 동작과 관련된 노드들은 VirMO/<x>/LockParams/Operations 노드 밑에 포함될 수 있다. 이 경우 잠금 동작과 관련된 노드인 Lock 노드는 위에서 설명된 VirMO/<x>/Operations/Lock 노드와 동일한 성질을 가지지만 VirMO/<x>/Operations 노드가 아니라 VirMO/<x>/LockParams/Operations 노드 아래에 위치될 수 있다. 마찬가지로, 잠금해제 동작과 관련된 노드인 Unlock 노드는 VirMO/<x>/Operations/Unlock 노드와 동일한 성질을 가지지만 VirMO/<x>/Operations 노드가 아니라 VirMO/<x>/LockParams/Operations 노드 아래에 위치될 수 있다.

[266] 도 9는 가상 머신 이미지 관리 오브젝트(Virtual Machine Image Management Object, VM Image MO)를 예시한 것이다. VM 이미지 MO는 VirMO 서버와 VirMO 클라이언트 간에 인터페이스를 제공하며, 이 인터페이스는 가상 머신 이미지들을 관리하는 데 사용될 수 있다. VM 이미지는 가상 머신을 생성하기

위한 설치 이미지(install image)로서 역할할 수 있다. 가상 머신 이미지 관리 오브젝트는 단말의 장치 관리 트리(즉, 단말 자체의 장치 관리 트리)에 포함될 수 있다.

[267] 도 9를 참조하면, VM 이미지 MO에 포함될 수 있는 노드들이 예시되어 있다. VM 이미지 노드에 포함되는 노드들에 대해 설명하면 다음과 같다.

[268] ■ VMImgMO

[269]

Tree Occurrence	Format	Min. Access Types
One	node	Get

[270] VMImgMO 노드는 VM 이미지 MO에 대한 루트 노드이다. DM 트리(tree)에서 VM 이미지 노드의 위치는 이 노드의 조상 엘리먼트(ancestor element)에 의해 정해진다. VM 이미지 MO에 대한 식별자(에, MOID)는 예를 들어 "urn:oma:mo:oma-vmimgmo:1.0"일 수 있다.

[271] ■ VMImgMO/<x>

[272]

Tree Occurrence	Format	Min. Access Types
ZeroOrMore	node	Get

[273] VMImgMO/<x> 노드는 특정 VM 이미지에 대한 파라미터들을 위한 플레이스홀더(placeholder)이다. 관리되는 VM 이미지들 각각에 대한 해당 파라미터들은 이 노드 아래에서 이용 가능하다.

[274] ■ VMImgMO/<x>/ImgID

[275]

Tree Occurrence	Format	Min. Access Types
One	chr	Get

[276] VMImgMO/<x>/ImgID 노드는 캐릭터(character) 값을 가지며 VM 이미지에 대한 식별자를 특정한다. 이 노드에 대한 값은 VirMO 클라이언트에 이해 할당될 수 있으며 그 값은 VM 이미지 MO 내에서 고유(unique)하다.

[277] ■ VMImgMO/<x>/Name

[278]

Tree Occurrence	Format	Min. Access Types
One	chr	Get

[279] VMImgMO/<x>/Name 노드는 VM 이미지의 이름을 특정한다. 예를 들어, VM 이미지의 이름은 "android_2.2_froyo"일 수 있다. 이 노드의 값은 VirMO 서버에 의해 설정될 수 있다.

[280] ■ VMImgMO/<x>/Description

[281]

Tree Occurrence	Format	Min. Access Types
ZeroOrOne	chr	Get

[282] VMImgMO/<x>/Description 노드는 VM 이미지에 대한 설명을 특정한다. VM 이미지에 대한 설명은 인간에 의해 판독가능할 수 있다.

[283] ■ VMImgMO/<x>/DownloadURI

[284]

Tree Occurrence	Format	Min. Access Types
ZeroOrOne	chr	Get

[285] VMImgMO/<x>/DownloadURI 노드는 VM 이미지를 다운로드하는 데 사용될 수 있는 URI(Uniform Resource Identifier)를 특정한다. 이 노드의 URI는 예를 들어 HTTP GET [RFC2616] 또는 Download Over the Air [DLOTA]와 같은 다운로드 메커니즘들에 이용될 수 있다.

[286] ■ VMImgMO/<x>/DownloadedLocation

[287]

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

[288] VMImgMO/<x>/DownloadedLocation 노드는 다운로드 완료된 VM 이미지의 로컬 경로(local path)를 특정한다. 예를 들어, VM 이미지의 로컬 경로는 "/sdcard/download/VirMO/images/android_2.2_froyo"일 수 있다. 이 노드의 값은 VMImgMO/<x>/State 노드의 값이 다운로드가 완료되었음을 나타내는 값(예, 2)인 경우 유효(valid)할 수 있다. 이 값의 포맷은 플랫폼마다 달라질 수 있다. 이 노드의 값은 VirMO 클라이언트에 의해 설정될 수 있다.

[289] ■ VMImgMO/<x>/State

[290]

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

[291] VMImgMO/<x>/State 노드는 VM 이미지의 현재 상태를 특정한다. 이 노드의 값은 VirMO 클라이언트에 의해 설정될 수 있다. 일 예로, 이 노드는 다음과 같은 값들을 가질 수 있다.

[292]

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.

[293] 예를 들어, 노드 값이 0일 경우, 다운로드는 아직 시작되지 않았으며 데이터가 수신되지 않았음을 나타낼 수 있다. 노드 값이 1일 경우, 다운로드가 시작되었음을 나타낼 수 있다. 노드 값이 2일 경우, 다운로드가 성공적으로 종료되었고 모든 데이터가 수신되었음을 나타낼 수 있다. 노드 값이 3일 경우, 다운로드가 실패하였고 0 바이트 이상의 데이터가 수신되었을 수 있음을 나타낼

수 있다. 이 외에도 다른 상태 및 값들의 조합이 가능하다. 또한, 이러한 값들은 오로지 예시에 불과하고 실시예에 따라 다른 값이 사용될 수 있으며, 각각의 값이 예시된 상태와 다른 상태를 나타낼 수 있다.

[294] ■ VMImgMO/<x>/ImgType

[295]

Tree Occurrence	Format	Min. Access Types
One	chr	Get

[296] VMImgMO/<x>/ImgType 노드는 VM 이미지의 콘텐츠 타입(또는 미디어 타입)을 특정한다. 노드 값은 MIME(Multipurpose Internet Mail Extensions) 미디어 타입일 수 있다. 이 노드의 값은 VirMO 서버에 의해 설정될 수 있다.

[297] ■ VMImgMO/<x>/Operations

[298]

Tree Occurrence	Format	Min. Access Types
One	node	Get

[299] VMImgMO/<x>/Operations 노드는 이 MO에 의해 지원되는 동작들에 대한 부모 노드이다.

[300] ■ VMImgMO/<x>/Operations/Download

[301]

Tree Occurrence	Format	Min. Access Types
One	null	Get

[302] VMImgMO/<x>/Operations/Download 노드는 VMImgMO/<x>/DownloadURI 노드를 이용하여 VM 이미지를 다운로드하기 위한 실행 명령(예, Exec command)으로 사용될 수 있다. VM 이미지를 완전히 다운로드한 후에, VirMO 클라이언트는 VMImgMO/<x>/DownloadedLocation 노드를 올바르게 설정해야 한다. 또한, VirMO 클라이언트는 다운로드 완료를 VirMO 서버에게 알리기 위해 VM Image Ready Alert를 VirMO 서버에게 보낼 수 있다.

[303] ■ VMImgMO/<x>/Operations/Delete

[304]

Tree Occurrence	Format	Min. Access Types
One	null	Get

[305] VMImgMO/<x>/Operations/Delete 노드는 VM 이미지를 제거하기 위한 실행 명령(예, Exec command)으로 사용될 수 있다. VM 이미지가 완전히 다운로드되거나 부분적으로 다운로드되는 경우 다운로드된 이미지는 제거될 수 있다. 이미지가 제거되는 경우 VMImgMO/<x> 아래의 해당 서브-트리가 제거될 수 있다.

[306] ■ VMImgMO/<x>/Ext

[307]

Tree Occurrence	Format	Min. Access Types
One	node	Get

[308] VMImgMO/<x>/Ext 노드는 VM 이미지 관련 정보를 저장하기 위한 확장용 노드이다. 이 확장용 노드는 단말 제조자(vendor) 특정 정보를 저장하는 데 사용될 수 있다.

[309] 이상에서, 도 9를 참조하여 VM 이미지 MO가 가질 수 있는 여러 노드들을 설명하였다. 하지만, 이러한 노드들은 오로지 예시일 뿐이며 실시예에 따라 다른 노드들을 추가로 포함할 수 있다. 또한, 실시예에 따라 상기 설명된 노드들 중 일부는 제외될 수 있다. 또한, 실시예에 따라 상기 노드들의 위치도 변경될 수 있다. 또한, 실시예에 따라 노드들의 이름도 변경될 수 있다. 예를 들어, VMImgMO/<x>/ImgType 노드는 VMImgMO/<x>/MediaType 노드로 지칭될 수 있다.

[310] VM 관리 동작 절차(VM Management Operation Procedures)

[311] VirMO 서버는 위에서 기술된 가상화 MO(VirMO)와 VM 이미지 MO(VM Image MO)를 이용하여 디바이스로 VM 이미지를 전달하고, 디바이스 내에 VM을 생성하거나 제거할 수 있다.

[312] 도 10은 VirMO 서버가 VM 이미지를 디바이스에 전달(deliver)하는 절차를 예시한 것이다.

[313] 도 10을 참조하면, S1002 단계에서 VirMO 서버가 VMImgMO 노드 아래에 서브-트리 생성을 위한 요청을 VirMO 클라이언트에게 보낼 수 있다.

[314] S1004 단계에서, VirMO 클라이언트는 VirMO 서버로부터 생성 요청을 수신한 후에 VMImgMO 노드 아래에 서브-트리를 생성할 수 있다. 그리고, VirMO 클라이언트는 VMImgMO/<x>/ImgID 노드의 값을 설정하고 VMImgMO/<x>/State 노드의 값을 VM 이미지가 다운로드되지 않았음을 나타내는 값(예, Not Downloaded 또는 0)으로 설정할 수 있다. VirMO 클라이언트는 생성 요청을 성공적으로 처리한 후에 그 결과(예, 상태 코드(status code))를 VirMO 서버로 보낼 수 있다.

[315] S1006 단계에서, VirMO 서버는 VM 이미지를 전달하는 데 사용되는 파라미터들을 적절히 설정할 수 있다. 그런 다음, VirMO 서버는 VMImgMO/<x>/Operations/Download 노드에 실행 명령(예, Exec command)를 보낼 수 있다.

[316] S1008 단계에서, VirMO 클라이언트는 실행 명령을 수신한 후에 VMImgMO/<x>/DownloadURI 노드에 저장된 URI로부터 VM 이미지 다운로드를 시작할 수 있다. 예를 들어, HTTP GET 또는 Download Over the Air[DLOTA]와 같은 대체 가능한 다운로드 메커니즘들이 VM 이미지 다운로드에 사용될 수 있다. VirMO 클라이언트는 VMImgMO/<x>/State 노드를 적절히 업데이트할 수 있다. VM 이미지가 완전히 다운로드되면, VirMO 클라이언트는 VMImgMO/<x>/DownloadedLocation 노드를 설정할 수 있다. VirMO 클라이언트는 동기식 보고 메커니즘(synchronous reporting mechanism) 또는 비동기식 보고 메커니즘(asynchronous reporting mechanism)을 이용하여 VirMO

서버에 응답을 보낼 수 있다.

[317] 도 11은 VirMO 서버가 디바이스에 가상 머신을 생성하는 절차를 예시한 것이다.

[318] 도 11을 참조하면, S1102 단계에서, VirMO 서버는 VirMO 노드 아래에 서브-트리 생성을 위한 요청을 VirMO 클라이언트에게 보낼 수 있다.

[319] S1104 단계에서, VirMO 클라이언트는 VirMO 서버로부터 요청을 수신한 후에 VirMO 노드 아래에 해당 서브-트리를 생성할 수 있다. VirMO 클라이언트는 VirMO/<x>/VMID 노드에 값을 할당할 수 있고 VirMO/<x>/State 노드의 값을 생성전 상태(NotCreated state)를 나타내는 값(예, 0)으로 설정할 수 있다. VirMO 클라이언트는 요청을 성공적으로 처리한 후에 그 결과(예, 상태 코드(status code))를 VirMO 서버로 보낼 수 있다.

[320] S1106 단계에서, VirMO 서버는 VirMO/<x>/CreateParams 노드를 적절히 설정할 수 있다. 그런 다음, VirMO 서버는 가상 머신을 생성하기 위해 실행 명령(예, Exec command)을 VirMO/<x>/Operations/Create로 보낼 수 있다.

[321] S1108 단계에서, VirMO 클라이언트는 실행 명령을 수신한 후에 VirMO/<x>/CreateParams 노드에 따라 새로운 가상 머신을 생성하고, VirMO/<x>/State 노드를 업데이트할 수 있다. VirMO 클라이언트는 동기식 보고 메커니즘 또는 비동기식 보고 메커니즘을 이용하여 VirMO 서버에 응답을 보낼 수 있다.

[322] 도 12는 VirMO 서버가 디바이스의 가상 머신을 제거하는 절차를 예시한 것이다.

[323] 도 12를 참조하면, S1202 단계에서, VirMO 서버는 VirMO/<x>/Operations/Delete 노드로 실행 명령(예, Exec command)을 보낼 수 있다.

[324] S1204 단계에서, VirMO 클라이언트는 실행 명령을 수신한 후에 디바이스 내의 해당 가상 머신을 제거할 수 있다. 이때, VirMO 클라이언트는 VirMO/<x> 노드 아래의 해당 서브-트리를 제거할 수 있다.

[325] 만일 위에서 설명된 제거중 상태(Deleting state)가 지원되는 경우, VirMO 클라이언트는 VM의 상태를 제거중 상태로 설정한 다음, 해당 VM과 VirMO/<x> 노드 아래의 해당 서브-트리를 제거할 수 있다.

[326] 도 13과 도 14는 사용자가 VM을 잠금해제할 때 VirMO 서버와 VirMO 클라이언트에서 각각 수행되는 절차를 예시한 것이다.

[327] 앞서 설명된 바와 같이, 디바이스 내의 가상 머신은 원격의 VirMO 서버에 의해 잠금될 수 있으며, 비인가(unauthorized) 액세스가 허용되지 않을 수 있다. 가상 머신을 잠근다는 것은 가상 머신이 임의의 비인가 액세스로부터 동작하지 않도록 만드는 것을 의미할 수 있다. 인가된(authorized) VirMO 서버는 VirMO 세션을 이용하여 가상 머신을 잠금해제할 수 있다. 혹은 사용자는 인증 비밀(authentication secret)(예, 패스워드 또는 코드) 입력을 통해 가상 머신을 잠금해제할 수 있다. VM을 잠금해제할 때, VirMO 서버는 보안 측면을 강화하기

위해 인증 정보(예, Auth 서브-트리)를 설정할 수 있다. 이러한 인증 정보는 사용자가 VM을 잠금해제할 수 있는지 여부를 판별하는 데 이용될 수 있다.

[328] 만일 VirMO/<x>/LockParams/UserUnlock 노드가 참(true)으로 설정되는 경우, VirMO 서버는 사용자를 인증하는 데 이용될 수 있는 VirMO/<x>/LockParams/UnlockAuth 서브-트리를 적절히 설정할 수 있다. 예를 들어, VirMO/<x>/LockParams/UnlockAuth/AuthType 노드에 의해 지시될 수 있는 인증 타입(authentication type)은 표 3과 같다.

[329] 표 3

[Table 3]

Value for “Auth/AuthType”	Descriptions
HTTP-BASIC	HTTP basic authentication done according to RFC 2617.
HTTP-DIGEST	HTTP digest authentication done according to RFC 2617.
BASIC	DM 'syncml:auth-basic' authentication as specified in [DMSEC].
DIGEST	DM 'syncml:auth-md5' authentication as specified in [DMSEC].
DIGEST-SHA256	DIGEST-SHA256 as specified in RFC6234
HMAC	DM 'syncml:auth-MAC' authentication as specified in [DMSEC].
X509	'syncml:auth-X509' authentication done according to [REPPRO].

[330] 표 3에서 [DMSEC]는 OMA 표준인 OMA-TS-DM_Security-V1_3, “OMA Device Management Security”을 의미하고, [REPPRO]는 OMA-TS-DM_RepPro-V1_3, “OMA Device Management Representation Protocol” 을 의미한다.

[331] VirMO/<x>/LockParams/UnlockAuth/AuthData 노드에 저장된 인증 데이터는 논스(nonce)처럼 동작하고 사용자 인증에 이용될 수 있다.

[332] 도 13을 참조하면, 사용자에 의해 VM이 잠금해제될 때 VirMO 서버가 잠금해제에 필요한 인증 정보를 단말에 설정해 주는 절차가 예시되어 있다.

[333] S1302 단계에서, VirMO 서버는 사용자 인증에 필요한 비밀(secret), 인증 데이터(AuthData), 및 인증 타입(AuthType)을 선택한다. 비밀(Secret)은 추후 사용자가 잠금해제하기 위해 입력할 정보이며, 이 비밀을 알고 있는 사용자 만이 VM을 잠금해제할 수 있다. 인증 데이터(AuthData)는 다이제스트(digest)를 계산하는 데 필요한 정보이다.

[334] S1304 단계에서, VirMO 서버는 비밀, 인증 데이터, 및 인증 타입에 따라 다이제스트 값을 계산할 수 있다. 다이제스트는 해쉬(hash) 함수에 의해 계산되며 digest = H(secret:AuthData)에 의해 산출될 수 있다. 즉 비밀에 인증 데이터를 이어 붙인 값을 해쉬 함수에 넣어서 다이제스트를 산출한다. 해쉬 함수는 인증 타입에 따라 달라질 수 있다.

[335] S1306 단계에서, VirMO 서버는 인증 데이터, 다이제스트, 및 인증 타입을 VirMO 클라이언트에게 보낼 수 있다. 가상화 MO(VirMO)에서 인증 데이터는 AuthData(예, VirMO/<x>/LockParams/UnlockAuth/AuthData)에 저장될 수 있고,

다이제스트는 AuthDigest(예, VirMO/<x>/LockParams/UnlockAuth/AuthDigest)에 저장될 수 있고, 인증 타입은 AuthType(예, VirMO/<x>/LockParams/UnlockAuth/AuthType)에 저장될 수 있다.

[336] 도 14를 참조하면, 사용자에 의해 VM이 잠금해제될 때 VirMO 클라이언트가 사용자로부터 비밀(secret)을 입력 받아 사용자 인증을 수행하는 절차가 예시되어 있다.

[337] S1402 단계에서, VirMO 클라이언트는 사용자로부터 특정 VM을 잠금해제하기 위한 비밀(sercret)을 입력받을 수 있다.

[338] S1404 단계에서, VirMO 클라이언트는 입력 받은 비밀(secret), 인증 데이터(AuthData)를 이용하여 인증 타입(AuthType)에 따라 다이제스트를 계산한다. 다이제스트의 계산은 VirMO 서버가 S1304 단계에서 수행한 방식과 동일할 수 있다. 예를 들어, digest = H(secret:AuthData)에 의해 산출될 수 있다. 즉, 사용자에게서 입력 받은 비밀에 AuthData 노드(예, VirMO/<x>/LockParams/UnlockAuth/AuthData)에 저장되어 있는 데이터를 이어 붙인 다음, 인증 타입(AuthType)에 따른 해쉬 함수를 이용해서 다이제스트를 계산한다.

[339] S1406 단계에서, S1404 단계에서 계산된 다이제스트가 AuthDigest 노드(예, VirMO/<x>/LockParams/UnlockAuth/AuthDigest)에 저장된 데이터와 일치하는지 확인할 수 있다. 일치하면 S1408 단계로 진행하고, 일치하지 않으면 S1414 단계로 진행한다.

[340] 사용자가 올바른 비밀을 입력한 것으로 확인되는 경우, S1408 단계에서, 사용자 인증이 성공되어 해당 VM이 잠금해제될 수 있다.

[341] S1410 단계에서, VirMO 클라이언트는 잠금해제 동작과 관련된 정보를 VirMO 서버에게 보고할 필요가 있는지 판단한다. 이 판단은 VirMO 서버가 미리 설정해준 정보를 통해 이루어질 수도 있고, VirMO 클라이언트가 자체적으로 판단할 수도 있다. VirMO 서버에게 보고가 필요하다고 판단하는 경우 S1412 단계로 진행하고, 필요하지 않는 경우 S1416 단계로 진행하여 절차는 종료된다.

[342] VirMO 서버에게 보고가 필요하다고 판단하는 경우, S1412 단계에서, VirMO 클라이언트는 VirMO 서버에게 일반 경보(Generic Alert)를 보낼 수 있다. 이 일반 경보는 경보 타입(alert type), 잠금해제 동작의 성공 여부, 시간 등의 정보를 포함할 수 있다.

[343] 만일 S1406 단계에서 사용자가 입력한 비밀이 올바르지 못한 것으로 확인되는 경우, S1414 단계에서 사용자 인증이 실패하여 해당 VM 잠금해제가 실패한다.

[344] 만일 S1410 단계에서 VirMO 서버에게 보고가 필요하지 않다고 판단되는 경우, 잠금해제 동작은 종료된다.

[345] VM 장치 관리(Device Management for Virtual Machine)

[346] 디바이스에 생성된 가상 머신은 운영 체제와 애플리케이션들이 그 안에서 동작할 수 있는 또 다른 독립적인 디바이스로서 고려될 수 있다. 가상 머신은

별도의 디바이스로서 원격 서버에 의해 관리될 수 있는데, 원격 서버에 의한 관리는 펌웨어 업데이트, 소프트웨어 설치/업데이트/제거, 원격 제어, 진단(diagnostic), 구성(configuration)을 포함할 수 있다. 이러한 유형의 관리는 VM 장치 관리(Device Management for Virtual Machine)로 지칭될 수 있다.

[347] VM 장치 관리를 위해 OMA(Open Mobile Alliance) DM(Device Management)에 정의된 장치 관리 기능들이 이용될 수 있다. OMA DM은 관리 오브젝트(Management Object, MO)를 이용해 장치 관리를 위한 기능들을 정의한다. 예를 들어, MO는 펌웨어를 업데이트하기 위한 FUMO("Firmware Update Management Object", OMA-TS-DM-FUMO-V1_0_2 참조), 소프트웨어 관리를 위한 SCOMO("Software Component Management Object", OMA-TS-DM-SCOMO-V1_0 참조), 원격 잠금을 위한 LAWMO("Lock and Wipe Management Object", OMA-TS-LAWMO-V1_0 참조) 등을 포함할 수 있다. 이외에도, MO는 앞으로 OMA DM에서 정의될 수 있는 임의의 관리 오브젝트를 포함할 수 있다. MO는 DM 서버와 DM 클라이언트 사이에 특정 기능을 위한 인터페이스로서의 역할을 수행할 수 있다.

[348] OMA DM에서 정의한 MO를 사용해 가상 머신을 관리하게 되면, 기존의 MO를 활용할 수 있는 장점이 있다. 즉, 가상 머신의 펌웨어 업데이트를 위해 별도의 기능을 정의할 필요가 없이 OMA DM의 FUMO를 적용하는 경우, 기존 기술이 재사용될 수 있어 가상 머신 관리가 효율적일 수가 있다. 이하에서는 MO를 활용하여 VM 장치 관리를 수행하는 방법을 설명한다.

[349] 도 15와 도 16은 별도의 DM 클라이언트를 이용하여 장치 관리를 수행하는 방법의 예를 예시한 것이다. 도 15는 원격 서버(예, DM 서버)가 장치 관리를 수행할 VM 내의 DM 클라이언트와 직접적으로 통신하여 장치 관리를 수행하는 직접 장치 관리(Direct Device Management)를 예시한 것이고, 도 16은 원격 서버(예, DM 서버)가 장치 관리를 수행할 VM 내의 DM 클라이언트와 간접적으로 통신하여 장치 관리를 수행하는 간접 장치 관리(Indirect Device Management)를 예시한 것이다.

[350] 도 15를 참조하면, 원격 서버(예, DM 서버(240) 또는 VirMO 서버(250))는 예를 들어 개인용 VM(336)과 엔터프라이즈 VM(334)를 생성하고 생성된 VM들에 대해 장치 관리를 제공할 수 있다. 직접 장치 관리의 경우, DM 클라이언트(210)가 각 VM(예, 개인용 VM(336), 엔터프라이즈 VM(334))에 설치될 필요가 있고, DM 서버(240)는 각 VM 내의 각 DM 클라이언트(1510, 1530)와 직접적으로 통신할 수 있다. DM 서버 입장에서 보면, 디바이스 내의 각 VM은 별도의 디바이스로서 보일 수 있다. 일부 실시예에서, 개인용 VM에 대해 장치 관리가 제공될 필요가 없을 수 있으므로 개인용 VM(336)에 DM 클라이언트(1510)가 설치되지 않을 수 있다. 또한, 도 15에서 타입 I 가상화를 기반으로 장치 관리를 수행하는 것으로 도시하였지만, 일부 실시예에서 타입 II 가상화를 기반으로 직접 장치 관리(Direct Device Management)가 제공될 수 있다.

[351] 도 15에 예시된 바와 같은 직접 장치 관리의 경우, VM 내의 DM 클라이언트는 DM 서버와 직접적으로 통신하기 위해 부트스트랩(bootstrap)되어야 한다. 또한, VM 장치 관리가 요구되는 VM마다 새로운 DM 클라이언트가 생성되기 때문에 VM에 새로운 디바이스 식별자(Device Identifier, Device ID)를 할당해야 한다. 디바이스 식별자는 단말의 고유한 정보(예, IMSI(International Mobile Station Identity), IMEI(International Mobile Equipment Identity), MSISDN(Mobile Station International ISDN Number) 등)에 의해 결정된다. 하지만, 종래에는 DM 클라이언트가 공장에서 수동으로 부트스트랩(즉, 공장 부트스트랩(Factory Bootstrap))되는 점을 감안하면, 도 15의 방법에서와 같이 동적으로 생성되는 DM 클라이언트를 부트스트랩하는 것이 어려울 수 있다. 즉, 도 15에서 관리 VM(332)은 통상 단말(또는 디바이스)에서 부트스트랩되어 있는 VM으로 볼 수 있고, 엔터프라이즈 VM(334)과 개인용 VM(336)은 동적으로 서버에 의해 생성된 것으로 볼 수 있다. 동적으로 생성된 VM에 관리 기능을 제공하고자 하는 경우, 엔터프라이즈 VM(334)을 위한 DM 클라이언트(1530)와 개인용 VM(336)을 위한 DM 클라이언트(1510)가 각각 부트스트랩되어야 한다. 하지만, 동적으로 생성된 VM에 대한 DM 클라이언트를 부트스트랩할 때 디바이스 식별자와 새로운 키를 할당해야 하므로 문제가 될 수 있다.

[352] 도 16을 참조하면, DM 서버(240)가 엔터프라이즈 VM(334)에 장치 관리를 제공하려고 하지만 엔터프라이즈 VM(334)의 DM 클라이언트(1230)는 DM 서버(240)와 직접적으로 통신할 수 없다. 이 경우, DM 서버(240)는 게이트웨이 MO(Gateway Management Object)를 이용하여 장치 관리를 제공할 수 있다.

[353] 이러한 간접 VM 장치 관리(indirect device management for VM)를 제공하기 위해 투명 모드(transparent mode)와 프록시 모드(proxy mode)가 이용될 수 있다. 투명 모드의 경우, 디바이스 내의 게이트웨이 MO 콤포넌트(1620)의 도움으로 DM 통지(notification)가 VM의 DM 클라이언트(1230)로 전달될 수 있다. DM 통지를 수신한 후, DM 클라이언트(1230)는 DM 서버(240)와 DM 세션을 개시할 수 있고, DM 서버(240)와 DM 클라이언트(1230) 사이에 VM 장치 관리가 수행될 수 있다. 투명 모드 동작의 경우에도, VM의 DM 클라이언트(1230)는 DM 서버(240)와 통신하기 위해 부트스트랩되어야 한다.

[354] 프록시 모드의 경우, 2개의 관련 DM 세션들이 개설(establish)될 수 있다. 하나는 DM 서버(240)와 DM 게이트웨이(DM 클라이언트(210)으로 동작) 사이에 개설될 수 있고, 다른 하나는 DM 게이트웨이(DM 서버(1640)로 동작)와 DM 클라이언트(1230) 사이에 개설될 수 있다. VM의 DM 클라이언트(1230)가 DM 서버(240)와 직접 통신하지 않기 때문에 DM 클라이언트(1230)는 디바이스 내에서 국지적으로(locally) 고유한 디바이스 식별자를 가질 수 있다. 프록시 모드 동작의 경우, VM의 DM 클라이언트(1230)가 DM 게이트웨이와 디바이스 내에서 통신하기 위해 부트스트랩되어야 한다.

[355] 도 17은 공유 DM 클라이언트를 이용하여 장치 관리를 수행하는 방법의 예를

예시한 것이다. 본 제안 방법은 VM 장치 관리를 필요로 하는 VM들에 각각 DM 클라이언트를 생성하지 않고 단말에 존재하는 하나의 DM 클라이언트를 이용하여 복수의 VM을 관리할 수 있다.

[356] 도 17을 참조하면, 관리 VM(332)은 디바이스(또는 단말)로 볼 수 있으므로 디바이스에 DM 클라이언트(210)가 존재한다. 개인용 VM(336)과 엔터프라이즈 VM(334)은 동적으로 VirMO 서버(250)에 의해 생성된 VM이며 이들 VM에 관리 기능을 제공하기 위해 별도의 DM 클라이언트(예, DM 클라이언트(1510), DM 클라이언트(1530))를 필요로 하지 않는다. 즉, 관리 VM(332)에 위치한 하나의 DM 클라이언트(210)를 이용해 다수의 VM을 관리하는 구조이다.

[357] 예를 들어, VirMO 서버(250)가 엔터프라이즈 VM(334)에 대한 장치 관리를 제공할 필요가 있을 때 관리 VM(332)의 DM 클라이언트(210)가 이용될 수 있다. VirMO 클라이언트(220)는 VirMO 서버(250)로부터의 장치 관리 명령을 다른 VM들로 포워딩할 수 있다. 원격 서버 입장에서 볼 때, 하나의 디바이스(예, 관리 VM(332))만이 존재한다.

[358] 더욱 구체적으로 예를 들면, 동적으로 생성된 VM들(엔터프라이즈 VM(334), 개인용 VM(336))은 각각 MO 클라이언트(1540)와 MO 클라이언트(1520)을 가질 수 있다. VirMO 서버(250)가 VirMO 클라이언트(220)에게 VM 관리를 위한 명령을 전달하면, 그 명령은 해당 VM에 위치한 MO 클라이언트(예, MO 클라이언트(1540) 또는 MO 클라이언트(1520))로 보내져 VM 관리가 수행된다. 도 17에서 DM 클라이언트(210)는 단말에 하나 존재할 수 있다. 반면에, 도 17에서 MO 클라이언트가 VM마다 존재하는 것으로 도시되어 있다. 하지만, 이는 예시일 뿐이며, MO 클라이언트는 VM 장치 관리가 필요한 VM마다 존재할 수도 있고, 디바이스(또는 단말)에 하나만 있을 수도 있다. 또한, 도 17의 방법은 타입 I 가상화를 기반으로 예시되어 있지만, 도 17의 방법은 타입 II 가상화를 기반으로 한 디바이스에도 적용될 수 있다.

[359] OMA DM을 통한 장치 관리를 위해 단말은 DM 트리를 저장할 필요가 있다. DM 트리는 단말이 지원하는 MO들을 포함할 수 있는데, 이러한 DM 트리를 통해 DM 서버는 DM 클라이언트에게 관리 명령을 전달할 수 있다. 마찬가지로, 도 17에서 장치 관리 명령을 다른 VM들로 포워딩하기 위해, VirMO 클라이언트는 다른 VM들에 대한 DM 트리를 가질 필요가 있다.

[360] 도 18은 VM 장치 관리(Device Management for Virtual Machine)를 위해 각 VM을 위한 DM 트리를 구성하는 방법을 예시한 것이다.

[361] 도 18을 참조하면, 동적으로 생성된 각 VM(예, VM1, VM2)을 위한 DM 트리가 가상화 관리 오브젝트(예, VirMO 노드) 아래의 특정 위치에 구성될 수 있다. 상기 특정 위치는 각 VM을 위한 노드 아래에 각 VM 장치 관리에 필요한 정보를 설정하기 위한 노드일 수 있다. 각 VM을 위한 DM 트리는 예를 들어 VirMO/<x>/DMTree 노드 밑에 위치할 수 있다. VirMO 서버는 VM에 제공하고자 하는 MO를 선택하여 VirMO/<x>/DMTree 노드 밑에 생성하고 관리 명령을

전달할 수 있다.

[362] 도 18의 예에서 VirMO 서버가 2개의 VM, VM1과 VM2를 생성했다고 가정한다. DM 서버가 VM1을 위한 장치 관리를 제공하고자 한다면, VM1을 위한 DM 트리(1810)가 VirMO/VM1/DMTree 노드 아래에 생성될 수 있다. VM 장치 관리를 제공하는 데 필요한 모든 관리 오브젝트들이 VirMO/VM1/DMTree 노드 아래에 생성될 수 있으며, 도 18의 예에서는 FUMO 서브-트리(1812)와 SCOMO 서브-트리(1814)가 생성되었다. 마찬가지로, DM 서버가 VM2를 위한 장치 관리를 제공하고자 한다면, VM2를 위한 DM 트리(1820)가 VirMO/VM2/DMTree 노드 아래에 생성될 수 있다. VM 장치 관리를 제공하는 데 필요한 모든 관리 오브젝트들이 VirMO/VM2/DMTree 노드 아래에 생성될 수 있으며, 도 18의 예에서는 LAWMO 서브-트리(1822)와 DiagMon 서브-트리(1824)가 생성되었다. 만일 DM 서버가 VM2를 위한 장치 관리를 제공하지 않는 경우에는 VirMO/VM2/DMTree 노드는 생성되지 않을 수 있다. 즉, DM 서버는 VM 장치 관리를 필요로 하는 VM에 대해서만 VirMO/<x>/DMTree 노드를 생성할 수 있다.

[363] VirMO 서버는 각 VM을 위한 DM 트리를 이용하여 관리 명령(management command)을 가상 머신으로 전달할 수 있다. 예를 들어, VM1의 펌웨어를 업데이트하기 위해 VirMO 서버는 VirMO/VM1/DMTree/FUMO/Update 노드에 실행 명령(예, Exec command)을 전달할 수 있다. 이 명령을 받은 VirMO 클라이언트는 VirMO/VM1/VMID를 통해 실행 명령(예, Exec command)이 VM1을 대상으로 하는 명령으로 해석할 수 있고, 따라서 VM1의 펌웨어를 업데이트할 수 있다. VirMO/<x> 노드의 이름을 VirMO/<x>/VMID와 동일하게 한 경우 VirMO/<x> 노드의 이름을 통해 대상 VM을 알 수도 있다. 비슷한 예로, 원격 제어를 통해 VM2를 잠금(Lock)하고자 할 때, VirMO 서버는 VirMO/VM2/DMTree/LAWMO/Operations/FullyLock에 실행 명령(예, Exec command)을 전달할 수 있다. 상기 예에서, VM1은 엔터프라이즈 VM이고 VM2는 개인용 VM일 수 있다.

[364] 도 18에 예시된 DM 트리 외에도 다른 MO를 위한 DM 트리가 생성될 수 있다. 다른 MO는 앞으로 OMA DM에서 정의될 수 있는 임의의 관리 오브젝트를 포함할 수 있다.

[365] 도 19는 VM 장치 관리를 수행하는 절차의 순서도를 예시한 것이다. 서버(예, VirMO 서버)와 클라이언트(예, VirMO 클라이언트)는 다음과 같은 절차를 통해 VM 장치 관리 명령(device management command for virtual machine)을 보내고 처리할 수 있다.

[366] 도 19를 참조하면, S1902 단계에서, 서버(예, VirMO 서버)는 가상 머신을 생성한 후에 관리 기능을 제공하고 싶은 관리 오브젝트(MO)를 선택하여 해당 VM 서브-트리의 장치 관리를 위한 노드(예, VirMO/<x>/DMTree 노드) 아래에 MO를 구성할 수 있다. 이 MO는 생성된 VM에 대해 장치 관리를 제공하는 데 필요할 수 있다. VirMO/<x>/DMTree 노드는 특정 VM에 속하는 DM 트리를 위한

루트 노드로서 역할할 수 있다. VirMO/<x>/DMTree 노드 아래에 MO 루트 노드가 위치될 수 있으며, MO 루트 노드는 각 MO의 타입을 구별하는 데 사용될 수 있는 관리 오브젝트 식별 정보(예, MOID (Management Object ID))를 가질 수 있다. 예를 들어, DevInfo와 같은 MO를 VirMO/<x>/DMTree 노드 밑에 구성하는 경우, DevID 정보는 장치 관리를 위한 노드에 저장된 VM 식별 정보(예, VMID)를 사용할 수 있다.

[367] S1904 단계에서, VirMO 서버는 MO가 특정 VM의 VirMO/<x>/DMTree 노드 밑에 구성되면, 해당 MO를 이용하여 VM을 위한 관리 명령을 전달할 수 있다. VirMO 서버는 관리 명령을 전달하려는 VM의 VirMO/<x>/DMTree 노드 밑에 위치한 MO를 대상으로 VM 관리 명령을 클라이언트(예, VirMO 클라이언트)에게 전달할 수 있다. 예를 들어, 엔터프라이즈 VM을 위해 SCOMO가 구성되는 경우, VirMO 서버는 엔터프라이즈 VM에 설치된 소프트웨어 콤포넌트들을 검색(retrieve)하기 위해 VirMO/EnterpriseVM/DMTree/SCOMO/Inventory/Deployed 노드를 타겟으로 가져오기 명령(예, Get command)을 보낼 수 있다.

[368] S1906 단계에서, VirMO 클라이언트는 VirMO 서버로부터 VM 관리 명령(예, DM 명령)을 수신하면, 클라이언트는 관리 명령(예, DM 명령)이 특정한 VM을 위한 장치 관리 명령인지 여부를 확인할 수 있다. 예를 들어, VirMO 클라이언트는 VM 관리 명령(예, DM 명령)이 타겟으로 하는 노드(예, 노드의 URI)를 기준으로 어느 VM을 대상으로 하는 관리 명령인지 판단할 수 있다. 이는 VM 장치 관리를 위한 MO가 특정 노드(예, VirMO/<x>/DMTree 노드) 밑에 위치하기 때문에 가능하다. 예를 들어, 관리 명령(예, DM 명령)이 VirMO/<x>/DMTree 아래의 노드를 타겟으로 하는 경우, VirMO 클라이언트는 VirMO/<x>/VMID 노드에 의해 가상 머신을 식별하고 식별된 가상 머신을 위한 장치 관리 명령이라고 여길 수 있다.

[369] 일 예로, 관리 명령(예, DM 명령)이 VirMO/VM1/DMTree/FUMO/Update 노드를 타겟으로 하는 경우, VirMO 클라이언트는 VirMO/VM1/VMID 노드를 통해 VM1을 대상으로 하는 펌웨어 업데이트(Firmware Update) 명령임을 알 수 있다. 다른 예로, 관리 명령(예, DM 명령)이 VirMO/VM1/DMTree/SCOMO/Inventory/Deployed 노드를 타겟으로 하는 경우 클라이언트(예, VirMO 클라이언트)는 이 관리 명령이 VM 식별 정보를 위한 노드(예, VirMO/VM1/VMID 노드)에 의해 식별되는 VM1에 설치된 소프트웨어 콤포넌트를 검색하기 위한 것임을 알 수 있다.

[370] 클라이언트(예, VirMO 클라이언트)는 특정 VM을 위한 장치 관리 명령을 처리한 후에 동기식 보고 메커니즘(synchronous reporting mechanism) 또는 비동기식 보고 메커니즘(asynchronous reporting mechanism)을 이용하여 서버(예, VirMO 서버)에게 응답할 수 있다.

[371] 도 20은 VM에서 발생된 일반 경보(Generic Alert)를 서버(예, VirMO 서버)로

보내고 처리하는 절차의 순서도를 예시한 것이다. VM에서 발생된 일반 경보에 대해 아래의 절차에 따라 클라이언트(예, VirMO 클라이언트)는 서버로 전송할 수 있고, 서버는 클라이언트로부터 수신한 일반 경보를 처리할 수 있다.

[372] 도 20을 참조하면, S2002 단계에서, VirMO 클라이언트는 VM 장치 관리를 위한 MO에서 일반 경보(Generic Alert)가 발생한 경우 소스(Source) 정보를 명시하여 서버(예, VirMO 서버)로 전송할 수 있다. 서버가 일반 경보가 발생한 VM을 구별하기 위해 소스(Source) 정보가 명시될 필요가 있다. 서버가 일반 경보를 수신하면, 서버는 소스 정보를 이용하여 해당 일반 경보가 어느 VM에서 발생했는지를 판단할 수 있다.

[373] 일 예로, 도 21은 소스 정보를 포함하는 일반 경보의 예를 예시한 것이다. 도 21의 예에서 서버(예, VirMO 서버)는 소스 정보를 이용하여 해당 일반 경보가 특정 VM(예, VM1)에서 발생했다고 판단할 수 있다. 또한, 도 21의 일반 경보는 서버에게 특정 VM(예, VM1)을 타겟으로 펌웨어 업데이트를 요청하는 것이므로 서버는 특정 VM(예, VM1)을 타겟으로 펌웨어 업데이트 관리 동작을 수행할 수 있다.

[374] 만일 소스 정보가 없다면, 일반 경보 수신자(예, VirMO 서버)는 해당 경보가 특정 VM에서 발생한 것이 아니라 단말 자체에서 발생했다고 판단할 수 있다. 또한, 소스 정보를 포함하는 경우에도, 일반 경보 수신자가 소스 정보를 이용하지 않거나 VM 관리 기능이 없어 소스 정보로부터 특정 VM에 대한 경보임을 알 수 없는 경우도 문제가 된다. 이를 위해 클라이언트(예, VirMO 클라이언트)는 타입 정보(예, Type)를 수정할 수 있다. 예를 들어, VM에서 발생한 일반 경보 타입 정보(예, Generic Alert Type)에 "-vm" 등과 같은 특정한 포스트픽스(postfix)를 추가할 수 있다. 이 경우, 이 타입 정보(예, Type)를 이해하지 못하는 일반 경보 수신자는 에러 코드(예, 415 “Unsupported media Type or Format”)를 전송할 수 있다. 도 22는 도 21의 일반 경보의 예에서 Type에 "-vm" 포스트픽스(postfix)를 붙인 예를 예시한 것이다.

[375] S2004 단계에서, 서버(예, VirMO 서버)는 수신된 일반 경보에 포함된 타입 및 소스 정보를 이용하여 일반 경보를 처리할 수 있다. 예를 들어, VM의 장치 관리를 위한 노드(예, VirMO/<x>/DMTree 노드) 밑에 놓여 있는 특정 VM을 위한 MO에서 일반 경보가 발생할 수 있다. 이 VM의 MO에서 발생된 경보는 일반 경보로 타입 정보(Type)과 소스 정보(Source)를 포함할 수 있다. 타입 정보는 일반 경보의 종류를 명시하는데, 예를 들어 "org.openmobilealliance.dm.firmwareupdate.devicerequest" 타입은 디바이스가 펌웨어 업데이트를 서버에게 요청하는 일반 경보임을 나타낸다. 소스 정보는 일반 경보가 발생한 MO와 관련된 주소 정보를 지정한다. 이 소스 정보를 통해, VirMO 서버는 일반 경보가 어느 VM에서 발생했는지 알 수 있다. 예를 들어, VirMO 서버가 수신한 일반 경보의 타입이 "org.openmobilealliance.dm.firmwareupdate.devicerequest"이고 소스 정보가

"./VirMO/VM1/MO/FUMO"인 경우, VirMO 서버는 VM1에서 해당 일반 경보가 발생했다는 것을 알 수 있다. 따라서, VirMO 서버는 VM1의 펌웨어 업데이트가 필요한지 여부를 조사할 수 있다. 만일 "-vm"과 같은 특별한 지시자가 포함된 경우, 서버(예, VirMO 서버)는 이 지시자를 삭제하여 원래의 일반 경보 타입을 알 수 있다.

[376] 사용자 상호 작용 경보의 처리 실시예(Embodiment of processing User Interaction Alert)

[377] 도 17에서 설명한 바와 같이, VM 장치 관리를 위해 각 VM 마다 DM 클라이언트를 별도로 두지 않고 VM을 위한 MO를 VM의 장치 관리를 위한 노드(예, VirMO/<x>/DMTree 노드) 밑에 위치시킬 경우 사용자 상호 작용 경보(User Interaction Alert, UI Alert)의 처리가 문제될 수 있다. DM 프로토콜의 사용자 상호 작용 경보는 사용자로부터 입력을 받는 등 사용자 상호 작용을 위해 정의되었다. 일 예로, 사용자에게 "Management in progress"라는 메시지를 적어도 10초 동안 보여주기 위해 UI 경보를 사용할 수 있으며, 이 UI 경보는 DM 클라이언트에게 전달될 수 있다.

[378] 도 23은 이러한 UI 경보를 예시한 것이다. 도 23의 예에서, 데이터 요소(예, <Data> element)에 포함되어 있는 숫자(예, 1100)는 이 경보가 UI 디스플레이 경보(Display Alert)임을 나타내고, 첫 번째 <Item><Data> 요소는 부가적인 정보(예, Minimum Display Time)를 나타내고, 두 번째 <Item><Data> 요소는 표시할 정보(예, Display Text)를 나타낼 수 있다.

[379] 기존 DM 프로토콜을 이용하여 특정 VM을 타겟으로 UI 경보를 전송하는 경우 DM 클라이언트는 UI 경보가 단말을 타겟으로 한 것으로 인식할 수 있다. 이는 기존 DM 프로토콜의 UI 경보는 VM과 관련된 정보를 포함하지 않기 때문이다. 따라서, 본 발명에서 VM 장치 관리를 위한 UI 경보를 구성하는 방법을 제안한다.

[380] 도 24는 VM을 타겟으로 하는 UI 경보를 구현하기 위한 실시예를 예시한 것이다.

[381] 도 24를 참조하면, 특정 VM을 타겟으로 하는 UI 경보를 위해 사용자 상호 작용에 관한 정보를 설정하기 위한 노드(예, VirMO/<x>/UI 노드)와 사용자 상호 작용에 관한 명령을 수신하기 위한 노드(예, ShowUI 노드)가 추가되었다. 도 24에서 사용자 상호 작용에 관한 정보를 설정하기 위한 노드(예, VirMO/<x>/UI 노드)는 UI 경보와 관련된 명령 정보(예, UICommands 노드)와 실행 결과(예, UIResults 노드)를 저장하는 서브-트리이다. 가상화 관리 오브젝트 동작과 관련된 노드인 VirMO/<x>/Operations 노드는 도 8을 참조하여 설명된 바와 같이 가상화 관리 오브젝트에 의해 지원되는 동작들에 대한 부모 노드(parent node)이다.

[382] 예를 들어, VirMO/<x>/Operations/ShowUI 노드는 VirMO 서버가 UI 경보를 통해 전송되는 UI 명령을 실행하기 위한 실행 명령(예, Exec command)으로 사용될 수 있다. VirMO 서버는 먼저 VirMO/<x>/UI/UICommands 노드에 UI 경보와 관련된 명령을 설정하고 VirMO/<x>/Operations/ShowUI 노드로 실행

명령을 전송할 수 있다. VirMO 클라이언트는 상기 노드로 실행 명령을 수신하면 VirMO/<x>/UI/UICommands 노드의 UI 명령을 실행하고 그 결과를 VirMO/<x>/UI/UIResults 노드에 저장할 수 있다.

[383] 도 25는 VM을 위한 사용자 상호 작용(UI)을 수행하는 절차를 예시한 것이다.

[384] 도 25를 참조하면, S2502 단계에서, 서버(예, VirMO 서버)는 UI 명령 정보를 설정하기 위한 노드(예, VirMO/<x>/UI/UICommands 노드)에 UI 경보 명령 정보를 설정한다. 예를 들어, UI 경보 명령 정보는 <Alert> 요소로 표현될 수 있으며, <Alert><Data> 요소는 UI 경보 코드 정보를 포함할 수 있다. 예를 들어, DM 표준 1.3에 정의된 가능한 UI 경보 코드는 1100 (디스플레이), 1101 (확인(Confirm) 또는 거절(Reject)), 1102 (텍스트 입력(Text Input)), 1103 (단일 선택(Single Choice)), 1104 (다중 선택(Multiple Choice))을 포함할 수 있다.

[385] S2504 단계에서, VirMO 서버는 UI 명령 정보를 설정하기 위한 노드(예, VirMO/<x>/UI/UICommands 노드)에 UI 경보 명령 정보를 설정한 후, UI 명령 수신을 위한 노드(예, VirMO/<x>/Operations/ShowUI 노드)에 실행 명령(예, Exec command)을 보낼 수 있다.

[386] S2506 단계에서, 클라이언트(예, VirMO 클라이언트)가 UI 명령 수신을 위한 노드(예, VirMO/<x>/Operations/ShowUI 노드)에 실행 명령(예, Exec command)을 수신하면, VirMO/<x>/UI/UICommands 노드에 저장된 UI 경보 명령을 수행할 수 있다. VirMO 클라이언트는 수행 결과를 나타내는 <Status> 요소를 VirMO/<x>/UI/UIResults 노드에 저장할 수 있다. 예를 들어, S2502 단계에서 VirMO 서버가 VirMO/<x>/UI/UICommands 노드에 도 26(a)와 같은 UI 경보를 설정할 수 있다. 사용자가 거절(Reject)하는 경우, 사용자가 입력한 정보는 <Status><Data> 요소 안에 담겨 VirMO/<x>/UI/UIResults 노드에 도 26(b)와 같이 저장될 수 있다.

[387] S2508 단계에서, VirMO 서버에 의해 VirMO/<x>/UI/UICommands 노드에 설정된 명령이 처리 완료되어 그 결과가 VirMO/<x>/UI/UIResults 노드에 저장되면, VirMO 클라이언트는 VirMO 서버에게 실행 명령이 완료되었음을 알린다.

[388] S2510 단계에서, VirMO 서버는 실행 명령이 완료되었다는 상태 코드(status code) 정보를 수신한 후 VirMO/<x>/UI/UIResults 노드에 저장된 결과 정보를 읽어 올 수 있다.

[389] S2508 단계와 S2510 단계는 동기식 보고 메커니즘을 사용한 경우의 예이지만, 비동기식 보고 메커니즘도 사용될 수 있다. 비동기식 보고 메커니즘을 사용할 경우, 예를 들어 S2508 단계에서 실행 명령(예, Exec 명령)이 처리되고 있음을 알리는 (202) Accepted for processing status code를 보낼 수 있다. 또한, 예를 들어 실행 명령 처리가 완료되었을 경우, VirMO 클라이언트는 일반 경보를 VirMO 서버에게 전송하여 실행 명령의 처리가 끝났음을 알릴 수 있다. 이 일반 경보를 수신한 후, VirMO 서버는 VirMO/<x>/UI/UIResults 노드에 저장된 결과 정보를

가져올 수 있다.

[390] 도 27은 본 발명이 적용될 수 있는 디바이스 및 서버를 예시한다.

[391] 도 27에 도시된 바와 같이, 본 발명이 적용될 수 있는 디바이스(110)는 프로세서(112)와 메모리(114)와 송수신 모듈(116)를 포함한다. 또한, 본 발명이 적용될 수 있는 서버(120)는 프로세서(122)와 메모리(124)와 송수신 모듈(126)를 포함한다.

[392] 메모리들(114, 124)은 프로세서(112)와 연결되고 도 5 내지 도 8에 도시된 방법들을 수행하기 위한 소프트웨어 프로그램 또는 명령어들 뿐만 아니라 프로세서(112)의 동작과 관련한 다양한 정보를 저장할 수 있다.

[393] 프로세서들(112, 122)은 메모리들(112, 122) 및 송수신 모듈들(116, 126)과 연결되고 이들을 제어한다. 구체적으로 프로세서들(112, 212)은 각각 메모리들(112, 122)에 저장된 소프트웨어 프로그램 또는 명령어들을 실행함으로써 방법들을 실행한다. 그리고 프로세서들(112, 212)은 송수신 모듈들(116, 126)을 통해 전술한 신호들을 송신 및/또는 수신한다.

[394] 전술된 실시예들 및 변형예들은 다양한 수단을 통해 구현될 수 있다. 예를 들어, 본 발명의 실시예들은 하드웨어, 펌웨어(firmware), 소프트웨어 또는 그것들의 결합 등에 의해 구현될 수 있다.

[395] 하드웨어에 의한 구현의 경우, 본 발명의 실시예들에 따른 방법은 하나 또는 그 이상의 ASICs(application specific integrated circuits), DSPs(digital signal processors), DSPDs(digital signal processing devices), PLDs(programmable logic devices), FPGAs(field programmable gate arrays), 프로세서, 콘트롤러, 마이크로 콘트롤러, 마이크로 프로세서 등에 의해 구현될 수 있다.

[396] 펌웨어나 소프트웨어에 의한 구현의 경우, 본 발명의 실시예들에 따른 방법은 이상에서 설명된 기능 또는 동작들을 수행하는 모듈, 절차 또는 함수 등의 형태로 구현될 수 있다. 소프트웨어 코드는 메모리 유닛에 저장되어 프로세서에 의해 구동될 수 있다. 상기 메모리 유닛은 상기 프로세서 내부 또는 외부에 위치하여, 이미 공지된 다양한 수단에 의해 상기 프로세서와 데이터를 주고 받을 수 있다.

[397] 예를 들어, 본 발명에 따른 방법은 컴퓨터 판독가능한 저장 매체(예를 들어, 내부 메모리, 플래쉬 메모리, 하드 디스크, 기타 등등)에 저장될 수 있고, 프로세서(예를 들어, 마이크로 프로세서)에 의해서 실행될 수 있는 소프트웨어 모듈(또는 프로그램) 내에 코드들 또는 명령어들로 구현될 수 있다.

[398] 본 발명의 실시예들을 구현하는 소프트웨어 모듈은 스크립트(script), 배치(batch), 또는 다른 실행가능한 파일들을 포함할 수 있다. 소프트웨어 모듈들은 디스크 드라이브와 같은 기계 판독가능한 또는 컴퓨터 판독가능한 저장 매체 상에 저장될 수 있다. 본 발명의 실시예에 따른 소프트웨어 모듈들을 저장하는 데 사용되는 저장 디바이스들은 예를 들어 자기 플로피 디스크, 하드 디스크, 또는 CD-ROM이나 CD-R과 같은 광학 디스크일 수 있다. 본 발명의

실시예에 따른 펌웨어나 하드웨어 모듈들을 저장하는 데 사용되는 저장 디바이스는 또한 반도체 기반의 메모리를 포함할 수 있으며, 이는 영구적으로, 탈착가능하게, 또는 원격으로 마이크로프로세서/메모리 시스템에 연결될 수 있다. 따라서, 상기 모듈들은 모듈의 기능들을 수행하는 컴퓨터 시스템을 구성하기 위해 컴퓨터 시스템 메모리 내에 저장될 수 있다. 다른 새롭고 다양한 유형의 컴퓨터 판독가능한 저장매체가 본 명세서에서 논의된 모듈들을 저장하는 데 사용될 수 있다. 게다가, 당해 기술분야의 통상의 기술자들은 기능들을 모듈들로 분리하는 것이 예시를 위한 목적이라는 것을 인식할 것이다. 대체가능한 실시예들은 다중 모듈들의 기능들을 단일 모듈로 병합할 수 있고, 또는 모듈들의 기능들을 대체가능하게 성분 분리할 수 있다. 예를 들어, 서브-모듈들을 호출하기 위한 소프트웨어 모듈들은 각 서브-모듈이 그것의 기능을 수행하고 제어를 직접적으로 다른 서브-모듈로 넘겨주도록 성분 분리될 수 있다.

[399] 이상에서 설명된 실시예들은 본 발명의 구성들과 특징들이 소정 형태로 결합된 것들이다. 각 구성 또는 특징은 별도의 명시적 언급이 없는 한 선택적인 것으로 고려되어야 한다. 각 구성 또는 특징은 다른 구성이나 특징과 결합되지 않은 형태로 실시될 수 있다. 또한, 일부 구성들 및/또는 특징들을 결합하여 본 발명의 실시예를 구성하는 것도 가능하다. 본 발명의 실시예들에서 설명되는 동작들의 순서는 변경될 수 있다. 어느 실시예의 일부 구성이나 특징은 다른 실시예에 포함될 수 있고, 또는 다른 실시예의 대응하는 구성 또는 특징과 교체될 수 있다. 특허청구범위에서 명시적인 인용 관계가 있지 않은 청구항들을 결합하여 실시예를 구성하거나 출원 후의 보정에 의해 새로운 청구항으로 포함시킬 수 있음은 자명하다.

[400] 본 발명은 본 발명의 정신 및 필수적 특징을 벗어나지 않는 범위에서 다른 특정한 형태로 구체화될 수 있다. 따라서, 상기의 상세한 설명은 모든 면에서 제한적으로 해석되어서는 안되고 예시적인 것으로 고려되어야 한다. 본 발명의 범위는 첨부된 청구항의 합리적 해석에 의해 결정되어야 하고, 본 발명의 등가적 범위 내에서의 모든 변경은 본 발명의 범위에 포함된다. 또한, 특허청구범위에서 명시적인 인용 관계가 있지 않은 청구항들을 결합하여 실시예를 구성하거나 출원 후의 보정에 의해 새로운 청구항으로 포함시킬 수 있다.

산업상 이용가능성

[401] 본 발명은 단말 및 서버 등과 같은 장치에 사용될 수 있다.

[402]

청구범위

[청구항 1] 복수의 가상 머신을 포함하는 단말에서 가상 머신을 위한 장치 관리를 수행하는 방법에 있어서,
특정 가상 머신을 생성하는 단계;
가상화 관리 오브젝트(virtualization management objection)에 상기 특정 가상 머신을 위한 장치 관리를 제공하는 데 필요한 적어도 하나의 관리 오브젝트를 구성하는 단계;
서버로부터 장치 관리 명령을 수신하는 단계;
상기 장치 관리 명령이 상기 특정 가상 머신을 위한 장치 관리 명령인지 여부를 확인하는 단계; 및
상기 장치 관리 명령이 상기 특정 가상 머신을 위한 장치 관리 명령인 경우 상기 장치 관리 명령을 처리하는 단계를 포함하되,
상기 가상화 관리 오브젝트는 상기 특정 가상 머신과 관련된 정보를 설정하기 위한 제1 노드를 포함하고, 상기 제1 노드 아래에 상기 특정 가상 머신을 위한 장치 관리에 필요한 정보를 설정하기 위한 제2 노드를 포함하며, 상기 적어도 하나의 관리 오브젝트는 상기 제2 노드 아래에 구성되는 방법.

[청구항 2] 제1항에 있어서,
상기 가상화 관리 오브젝트는 상기 제1 노드 아래에 상기 특정 가상 머신에 대한 식별 정보를 설정하기 위한 노드를 더 포함하고,
상기 확인하는 단계는 상기 장치 관리 명령이 상기 제2 노드 아래에 위치한 노드를 타겟으로 하는 경우 상기 장치 관리 명령은 상기 식별 정보에 의해 식별되는 가상 머신을 위한 것으로 결정하는 것을 포함하는 방법.

[청구항 3] 제1항에 있어서,
상기 적어도 하나의 관리 오브젝트는 상기 제2 노드 아래에 구성되는 루트 노드를 가지고, 상기 루트 노드는 관리 오브젝트 식별 정보를 포함하며,
상기 적어도 하나의 관리 오브젝트는 상기 관리 오브젝트 식별 정보에 의해 구별되는 방법.

[청구항 4] 제1항에 있어서,
상기 가상화 관리 오브젝트는 단말의 장치 관리 트리에 포함되는 방법.

[청구항 5] 제2항에 있어서,
상기 특정 가상 머신에 대한 식별 정보는 단말에 의해 주어지고 상기 가상화 관리 오브젝트 내에서 상기 특정 가상 머신을 고유하게 지시하는 방법.

[청구항 6] 제1항에 있어서,
상기 가상화 관리 오브젝트는 상기 제1 노드 아래에 사용자 상호 작용에 관한 정보를 설정하기 위한 제3 노드와 사용자 상호 작용에 관한 명령을 수신하기 위한 제4 노드를 포함하며,
상기 장치 관리 명령을 수신하는 단계는 상기 제4 노드를 통해 실행 명령을 수신하는 것을 포함하고,
상기 장치 관리 명령을 처리하는 단계는 상기 제3 노드에 설정된 상기 사용자 상호 작용에 관한 명령 정보를 실행하는 것을 포함하는 방법.

[청구항 7] 가상화 관리 오브젝트와 가상 머신 이미지 관리 오브젝트를 가지는 단말에서 가상 머신을 생성하는 방법에 있어서, 상기 가상 머신 이미지 관리 오브젝트는 복수의 가상 머신들의 가상 머신 이미지 식별 정보가 각각 설정된 복수의 노드들을 포함하며,
서버로부터 실행 명령을 수신하는 단계; 및
상기 실행 명령이 상기 가상화 관리 오브젝트의 가상 머신 생성 동작과 관련된 제1 노드로 전송된 경우, 상기 가상화 관리 오브젝트의 제2 노드에 설정된 가상 머신 생성 파라미터에 따라 특정 가상 머신을 생성하는 단계를 포함하되,
상기 제2 노드에 설정된 가상 머신 생성 파라미터는 상기 특정 가상 머신을 생성하는 데 사용되는 가상 머신 이미지를 특정하는 제1 정보를 포함하고, 상기 제1 정보는 상기 복수의 노드들에 설정된 가상 머신 이미지 식별 정보들 중 하나인 방법.

[청구항 8] 제7항에 있어서,
상기 가상 머신 생성 파라미터는 상기 생성된 특정 가상 머신의 상태를 지시하는 제2 정보를 더 포함하고,
상기 제2 정보가 제1 값인 경우, 상기 생성된 특정 가상 머신은 실행 중인 상태에 있고,
상기 제2 정보가 제2 값인 경우, 상기 생성된 특정 가상 머신은 파워 오프 상태에 있는 방법.

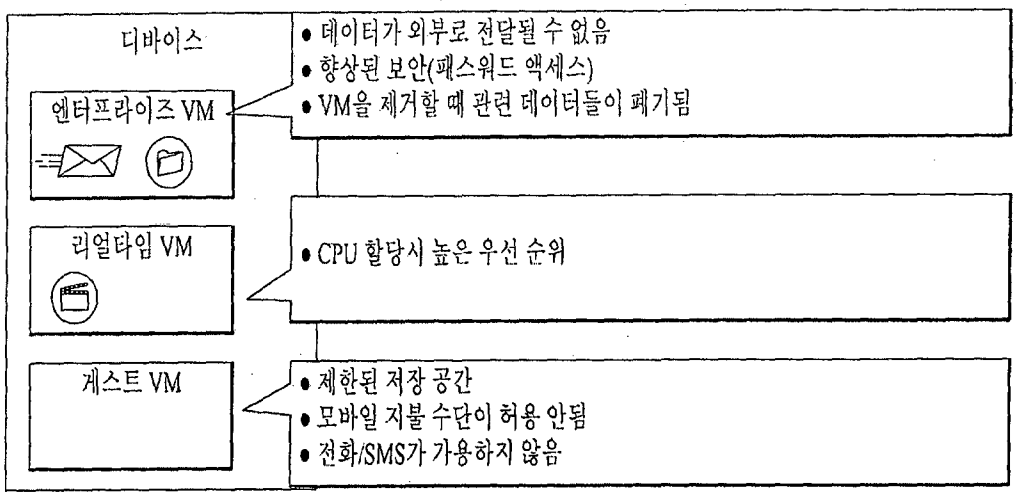
[청구항 9] 제7항에 있어서,
상기 가상 머신 이미지 관리 오브젝트는 단말의 장치 관리 트리에 포함되는 방법.

[청구항 10] 제7항에 있어서,
상기 가상 머신 이미지 식별 정보는 단말에 의해 주어되고 상기 가상 머신 이미지 관리 오브젝트 내에서 특정 가상 머신 이미지를 고유하게 지시하는 방법.

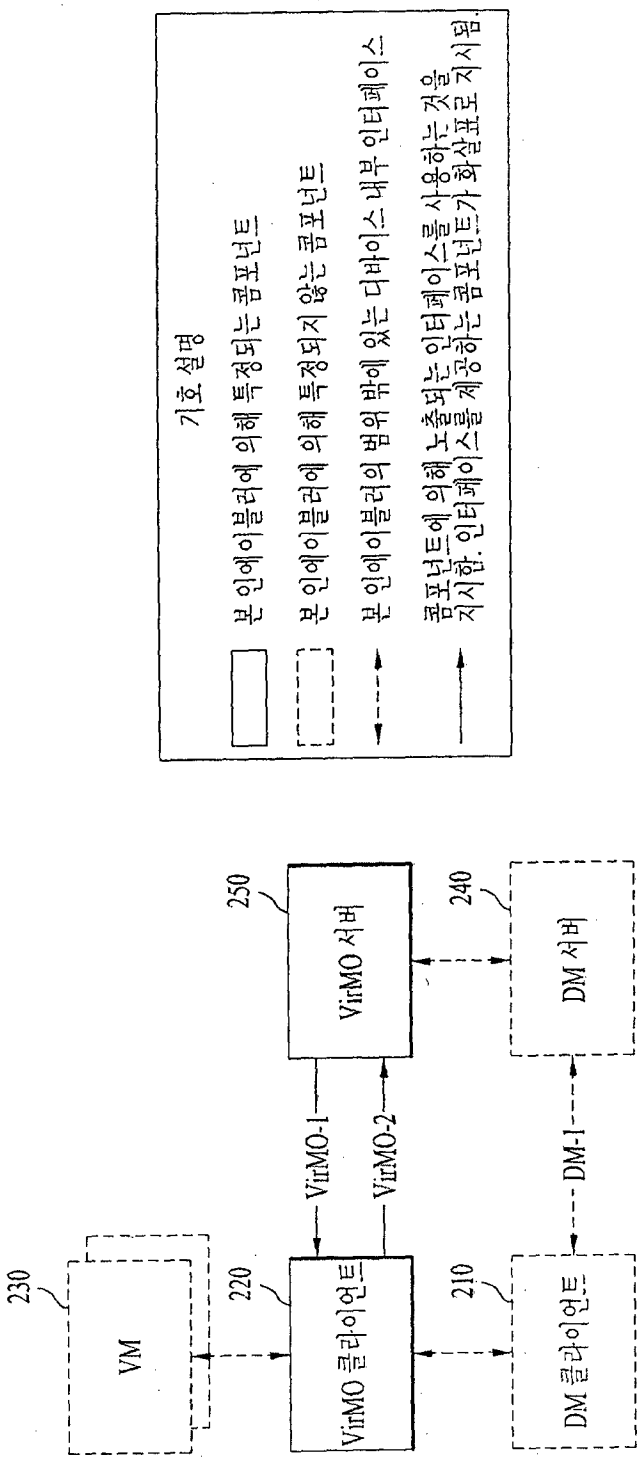
[청구항 11] 제7항에 있어서,
동기식 보고 메커니즘 또는 비동기식 보고 메커니즘을 이용하여

상기 서버에 응답하는 단계를 더 포함하는 방법.

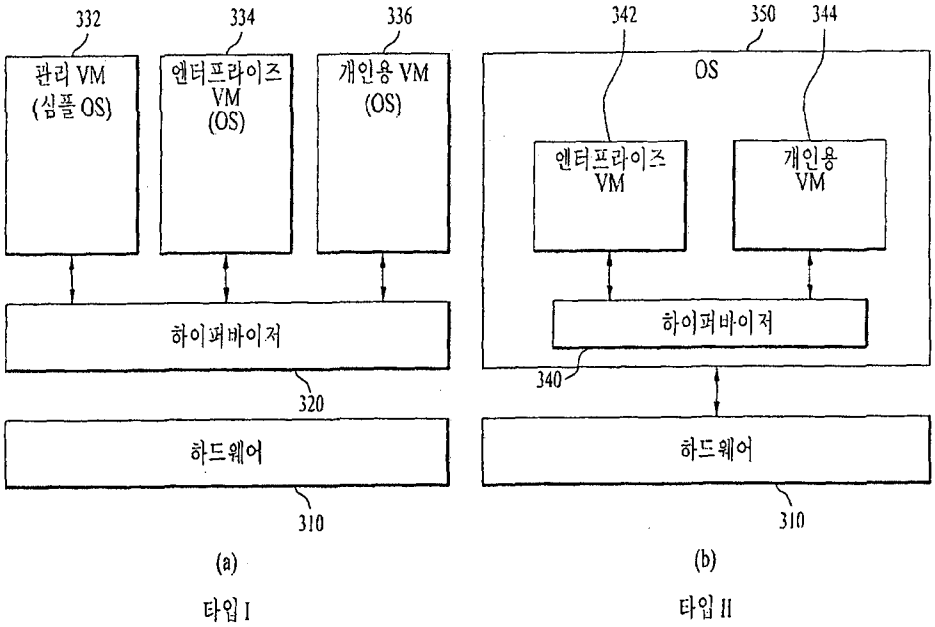
[도 1]


디바이스
엔터프라이즈 VM
리얼타임 VM
게스트 VM
• 데이터가 외부로 전달될 수 없음
• 향상된 보안(패스워드 액세스)
• VM을 제거할 때 관련 데이터들이 폐기됨
• CPU 할당시 높은 우선 순위
• 제한된 저장 공간
• 모바일 지불 수단이 허용 안됨
• 전화/SMS가 가용하지 않음

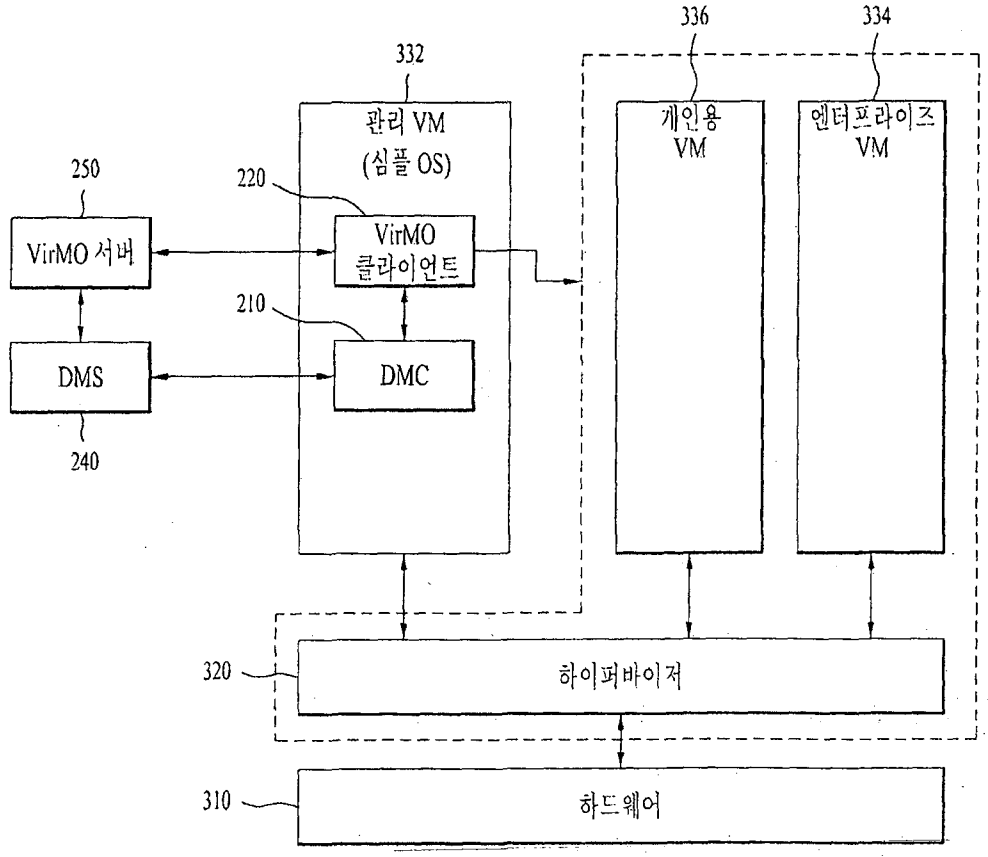
[도 2]


기호 설명
본 인에이블러에 의해 특정되는 콤포넌트
본 인에이블러에 의해 특정되지 않는 콤포넌트
본 인에이블러의 범위 밖에 있는 디바이스 내부 인터페이스
콤포넌트에 의해 노출되는 인터페이스를 사용하는 것을 지시함. 인터페이스를 제공하는 콤포넌트가 화살표로 지시됨.
VM
VirMO 클라이언트
VirMO 서버
DM 클라이언트
DM 서버
VirMO-1
VirMO-2
DM-1
230
220
210
250
240

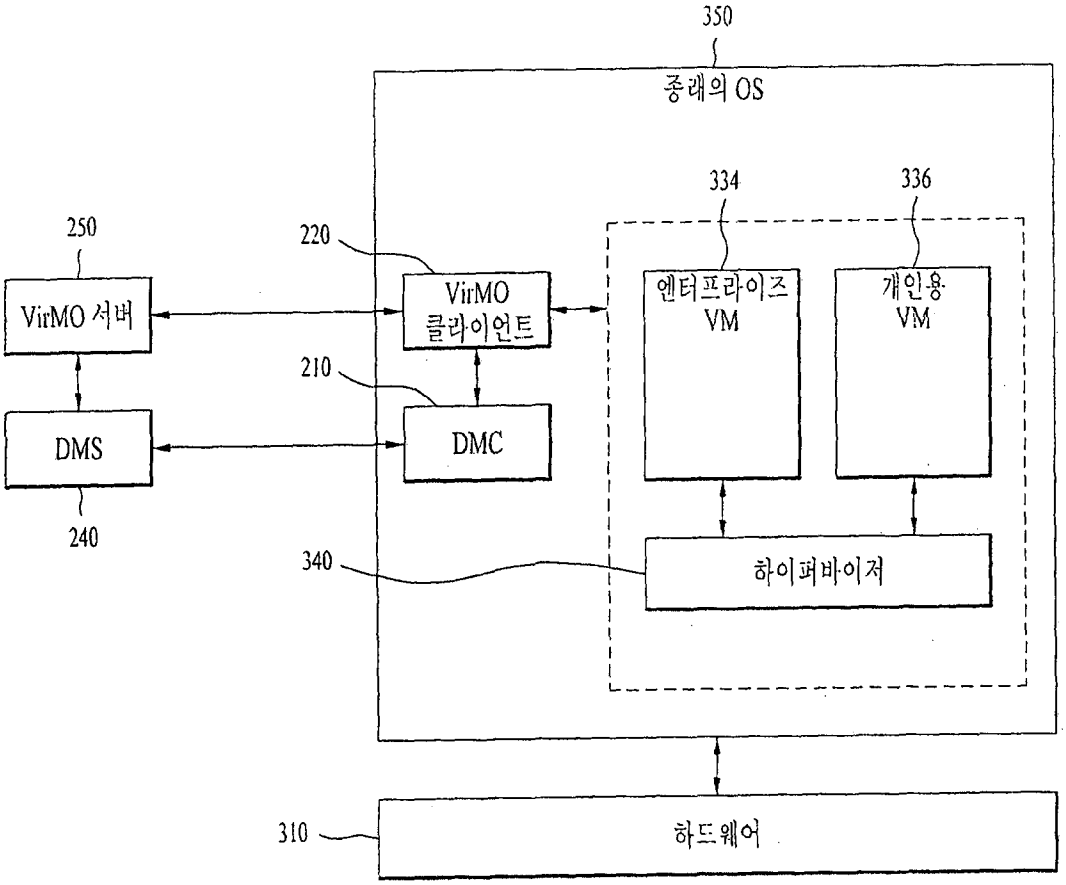
[도 3]


332
334
336
관리 VM
(심플 OS)
엔터프라이즈
VM
(OS)
개인용 VM
(OS)
하이퍼바이저
320
하드웨어
310
(a)
타입 I
342
350
344
OS
엔터프라이즈
VM
개인용
VM
하이퍼바이저
340
하드웨어
310
(b)
타입 II

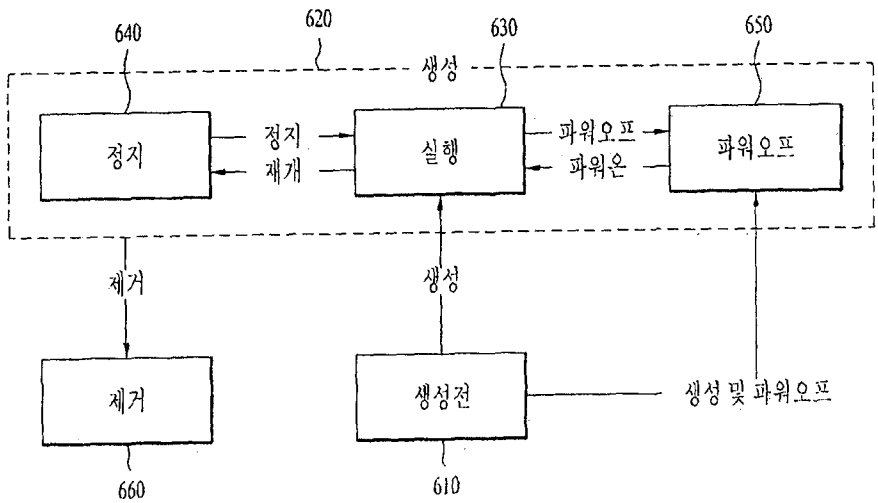
[도 4]


336
334
332
관리 VM
(심플 OS)
개인용
VM
엔터프라이즈
VM
250
220
VirMO 서버
VirMO
클라이언트
210
DMS
DMC
240
320
하이퍼바이저
310
하드웨어

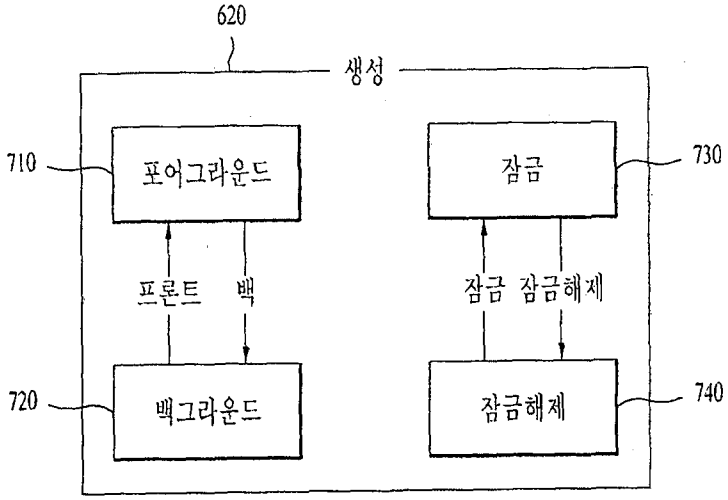
[도 5]


350
종래의 OS
334
336
250
220
VirMO 서버
VirMO
클라이언트
엔터프라이즈
VM
개인용
VM
210
DMS
DMC
240
340
하이퍼바이저
310
하드웨어

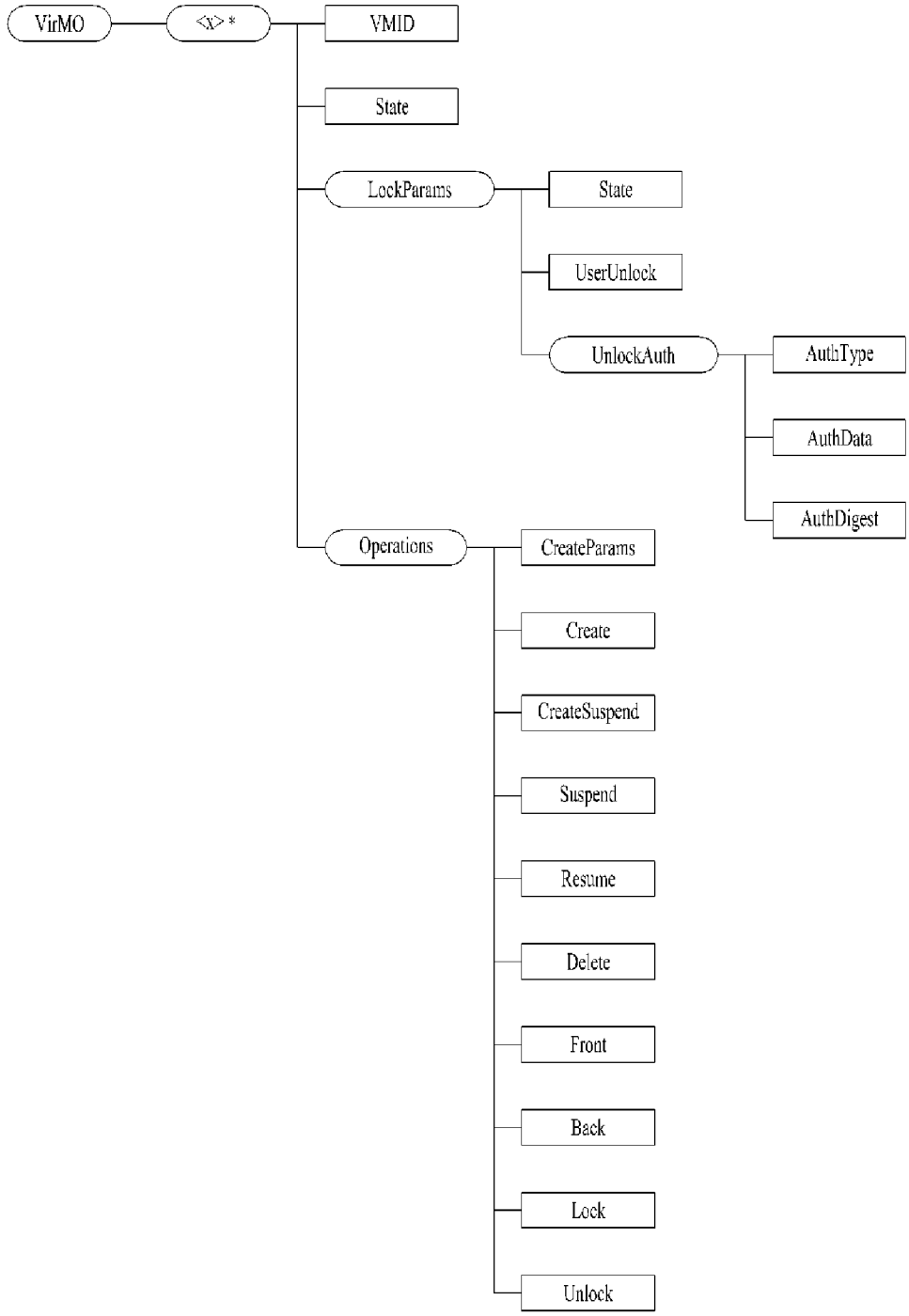
[도 6]


640
620
630
650
생성
정지
정지
재개
실행
파워오프
파워온
파워오프
제거
생성
제거
생성전
생성 및 파워오프
660
610

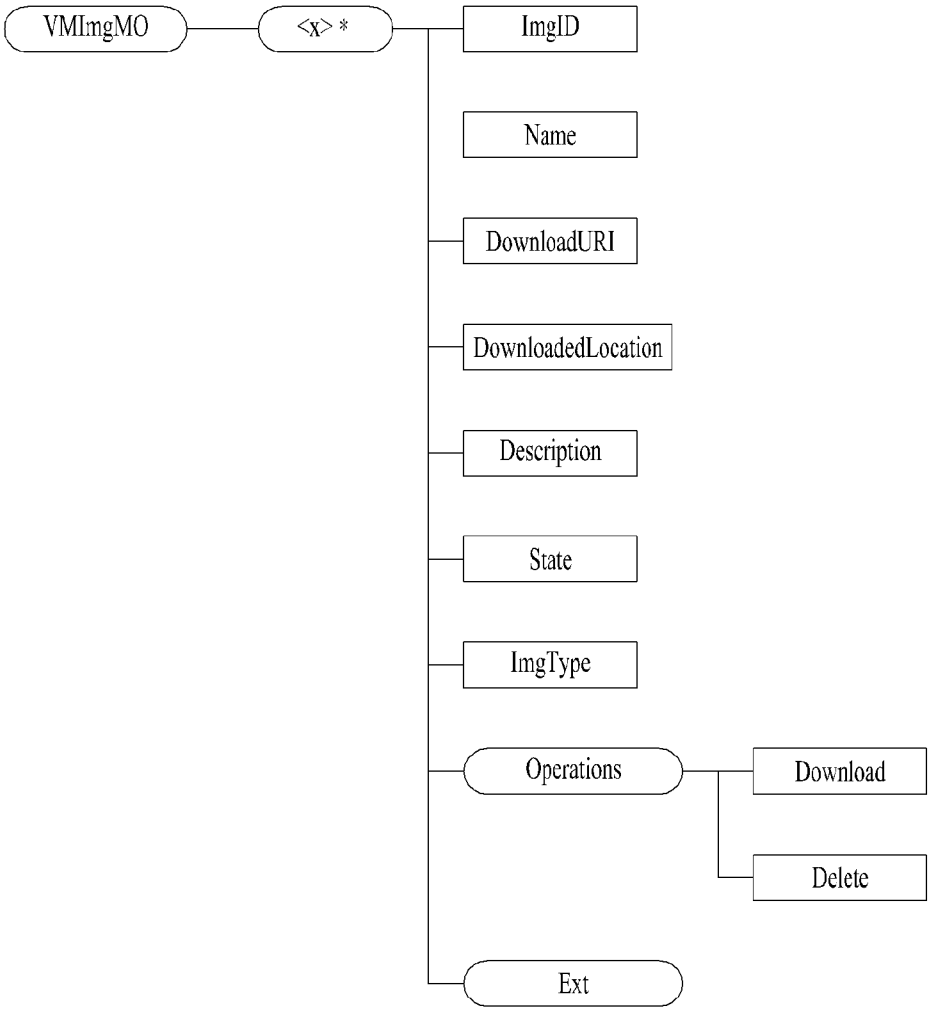
[도 7]



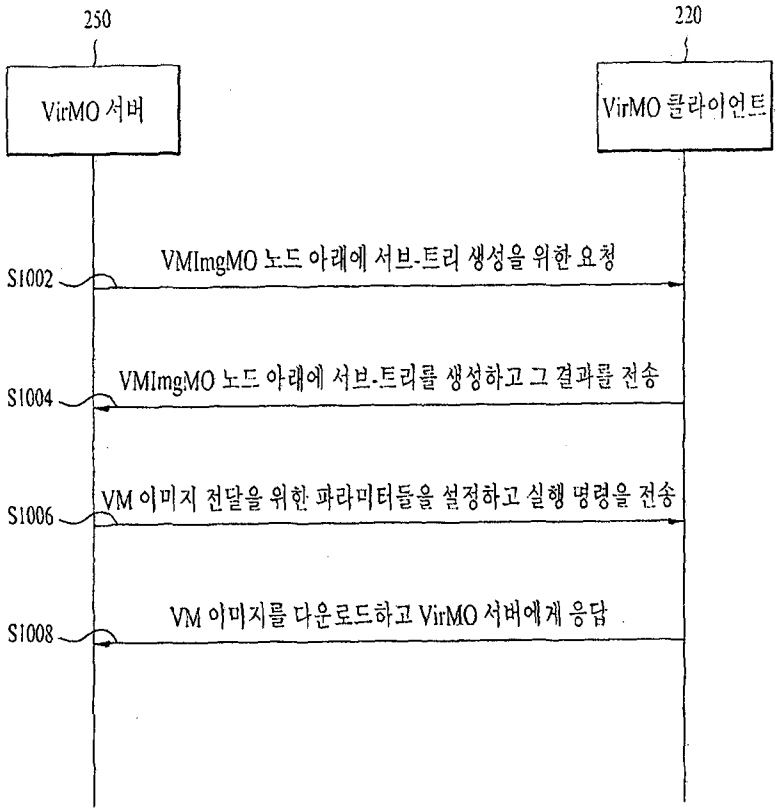
[Fig. 8]


VirMO
<X> *
VMID
State
LockParams
State
UserUnlock
UnlockAuth
AuthType
AuthData
AuthDigest
Operations
CreateParams
Create
CreateSuspend
Suspend
Resume
Delete
Front
Back
Lock
Unlock

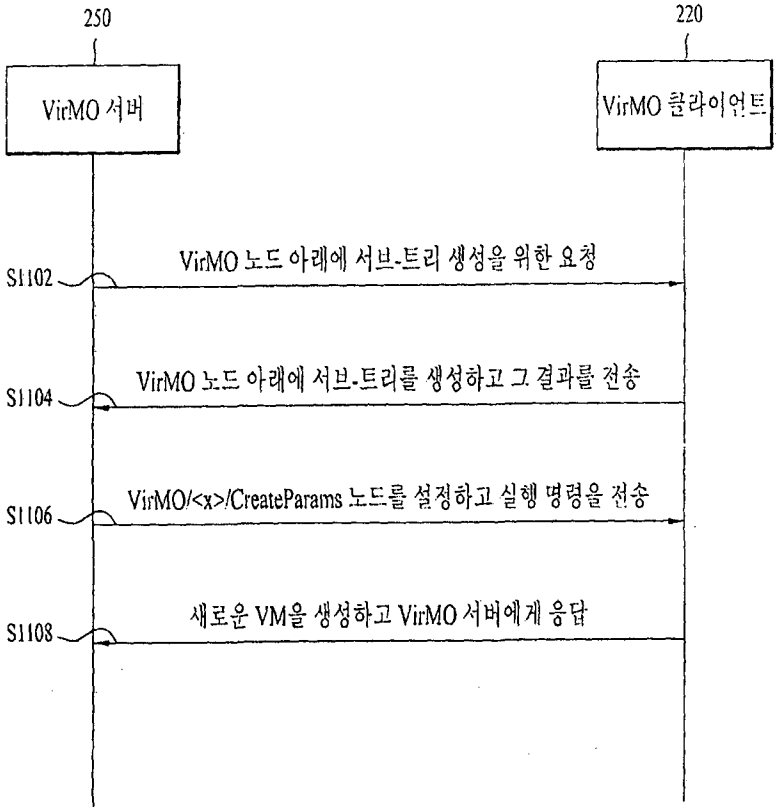
[Fig. 9]


VMImgMO
<x> *
ImgID
Name
DownloadURI
DownloadedLocation
Description
State
ImgType
Operations
Download
Delete
Ext

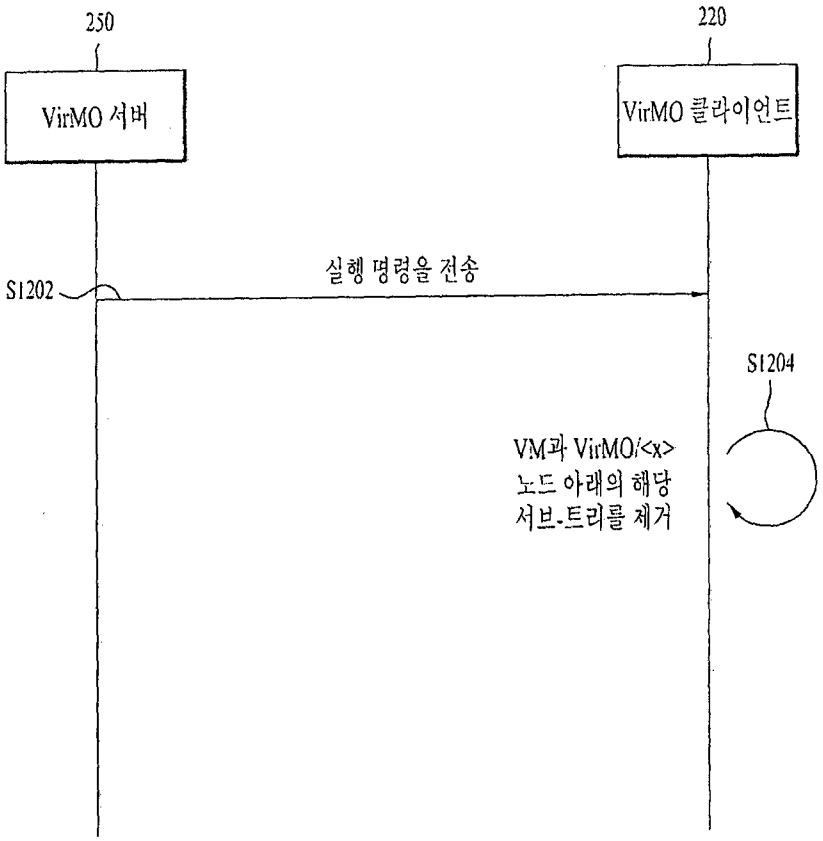
[도 10]


250
220
VirMO 서버
VirMO 클라이언트
S1002
VMImgMO 노드 아래에 서브-트리 생성을 위한 요청
S1004
VMImgMO 노드 아래에 서브-트리를 생성하고 그 결과를 전송
S1006
VM 이미지 전달을 위한 파라미터들을 설정하고 실행 명령을 전송
S1008
VM 이미지를 다운로드하고 VirMO 서버에게 응답

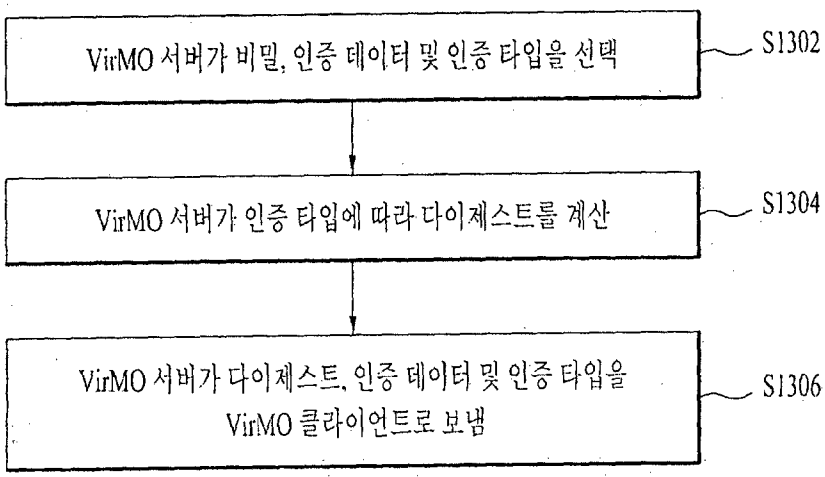
[도 11]


250
220
VirMO 서버
VirMO 클라이언트
S1102
VirMO 노드 아래에 서브-트리 생성을 위한 요청
S1104
VirMO 노드 아래에 서브-트리를 생성하고 그 결과를 전송
S1106
VirMO/<x>/CreateParams 노드를 설정하고 실행 명령을 전송
S1108
새로운 VM을 생성하고 VirMO 서버에게 응답

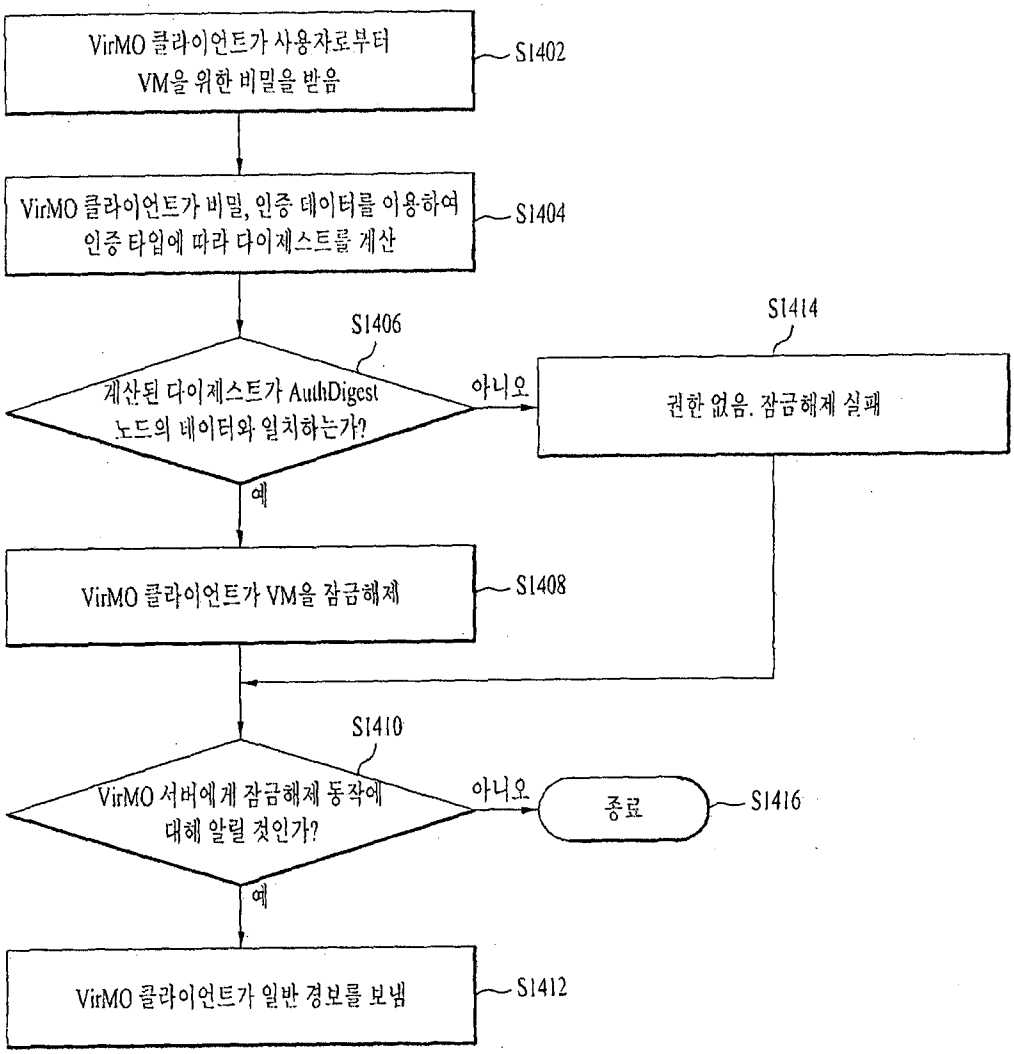
[도 12]


250
220
VirMO 서버
VirMO 클라이언트
S1202
실행 명령을 전송
S1204
VM과 VirMO/<x>
노드 아래의 해당
서브-트리를 제거

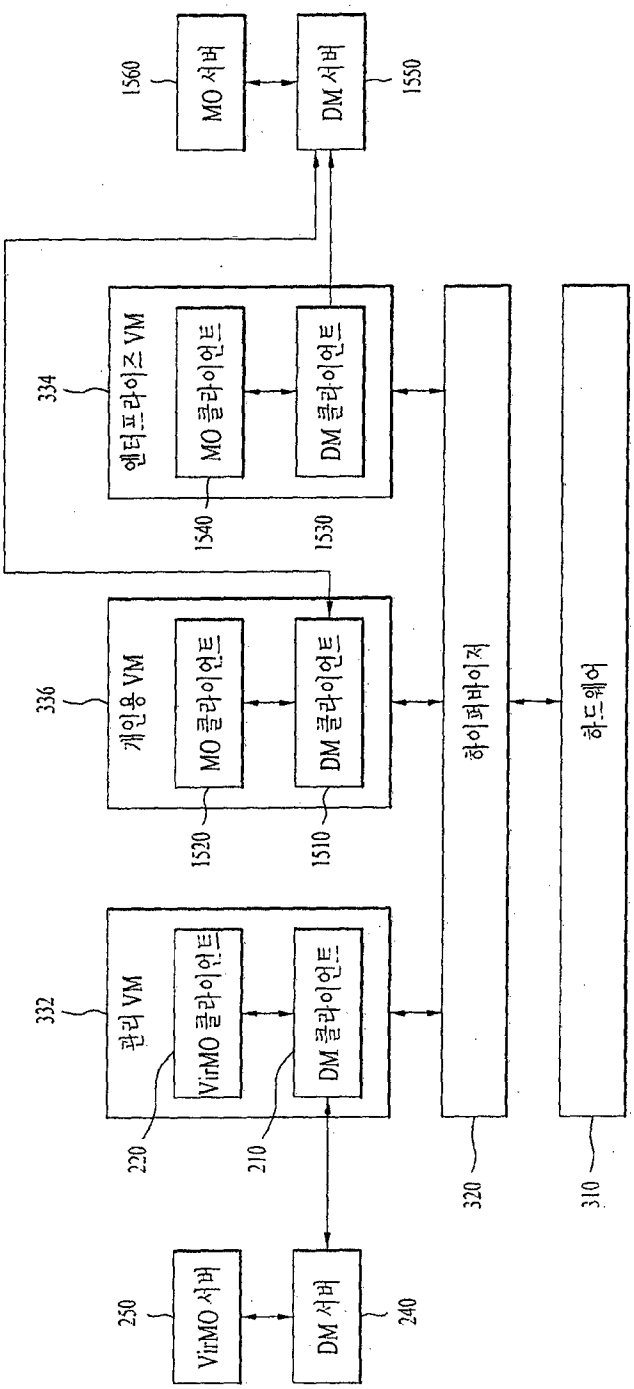
[도 13]


VirMO 서버가 비밀, 인증 데이터 및 인증 타입을 선택
S1302
VirMO 서버가 인증 타입에 따라 다이제스트를 계산
S1304
VirMO 서버가 다이제스트, 인증 데이터 및 인증 타입을
VirMO 클라이언트로 보냄
S1306

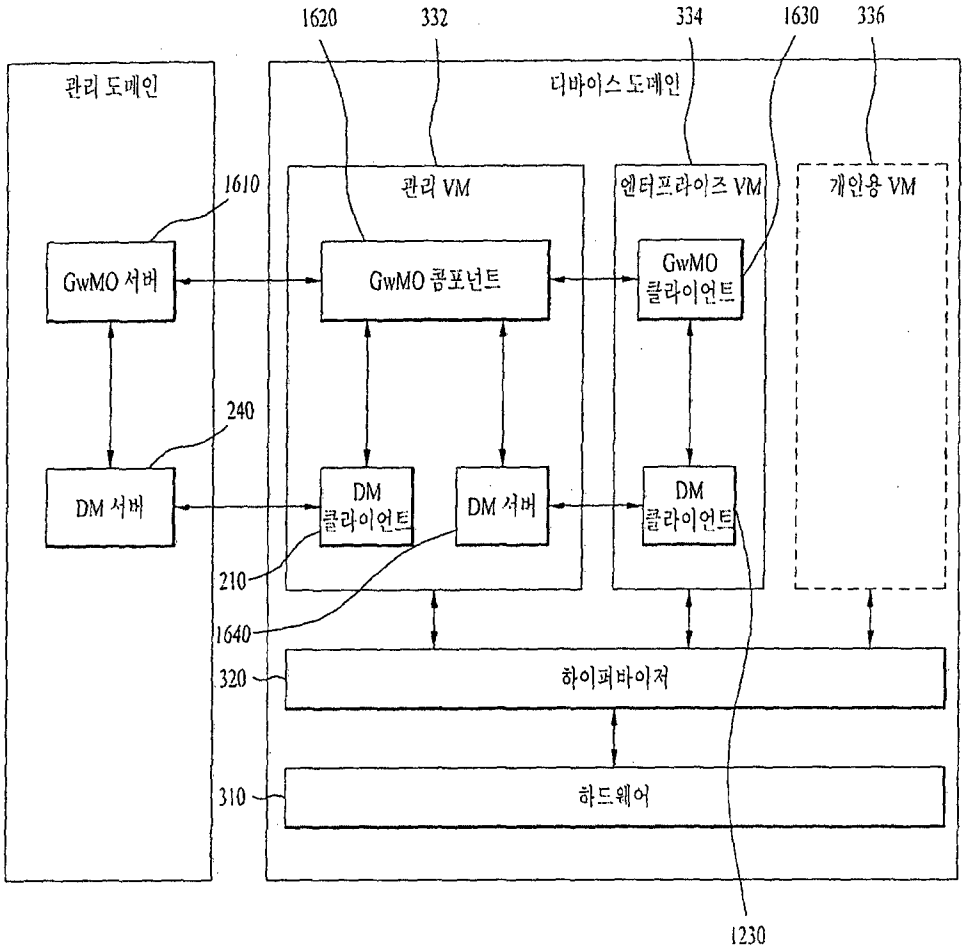
[도 14]


VirMO 클라이언트가 사용자로부터 VM을 위한 비밀을 받음
S1402
VirMO 클라이언트가 비밀, 인증 데이터를 이용하여 인증 타입에 따라 다이제스트를 계산
S1404
S1406
계산된 다이제스트가 AuthDigest 노드의 데이터와 일치하는가?
아니오
S1414
권한 없음. 잠금해제 실패
예
VirMO 클라이언트가 VM을 잠금해제
S1408
S1410
VirMO 서버에게 잠금해제 동작에 대해 알릴 것인가?
아니오
종료
S1416
예
VirMO 클라이언트가 일반 경보를 보냄
S1412

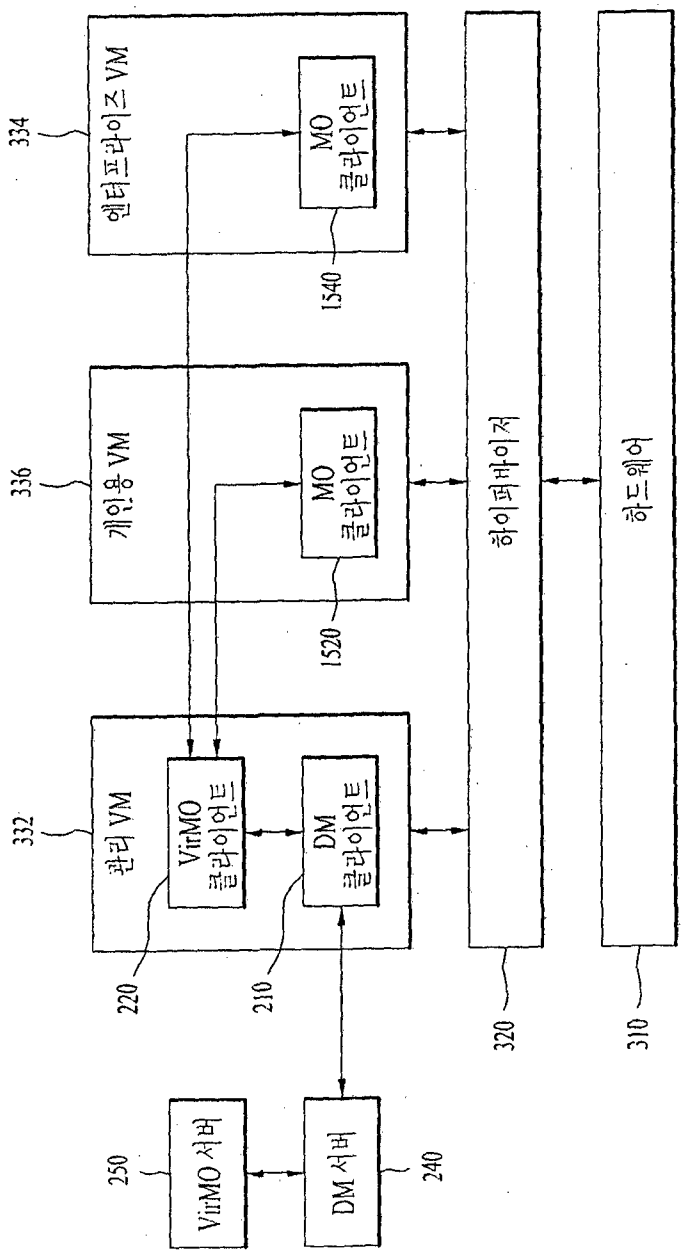
[도 15]


MO 서버
1560
DM 서버
1550
엔터프라이즈 VM
334
MO 클라이언트
DM 클라이언트
1540
1530
게인용 VM
336
MO 클라이언트
DM 클라이언트
1520
1510
하이퍼바이저
320
하드웨어
310
관리 VM
332
VirMO 클라이언트
DM 클라이언트
220
210
VirMO 서버
250
DM 서버
240

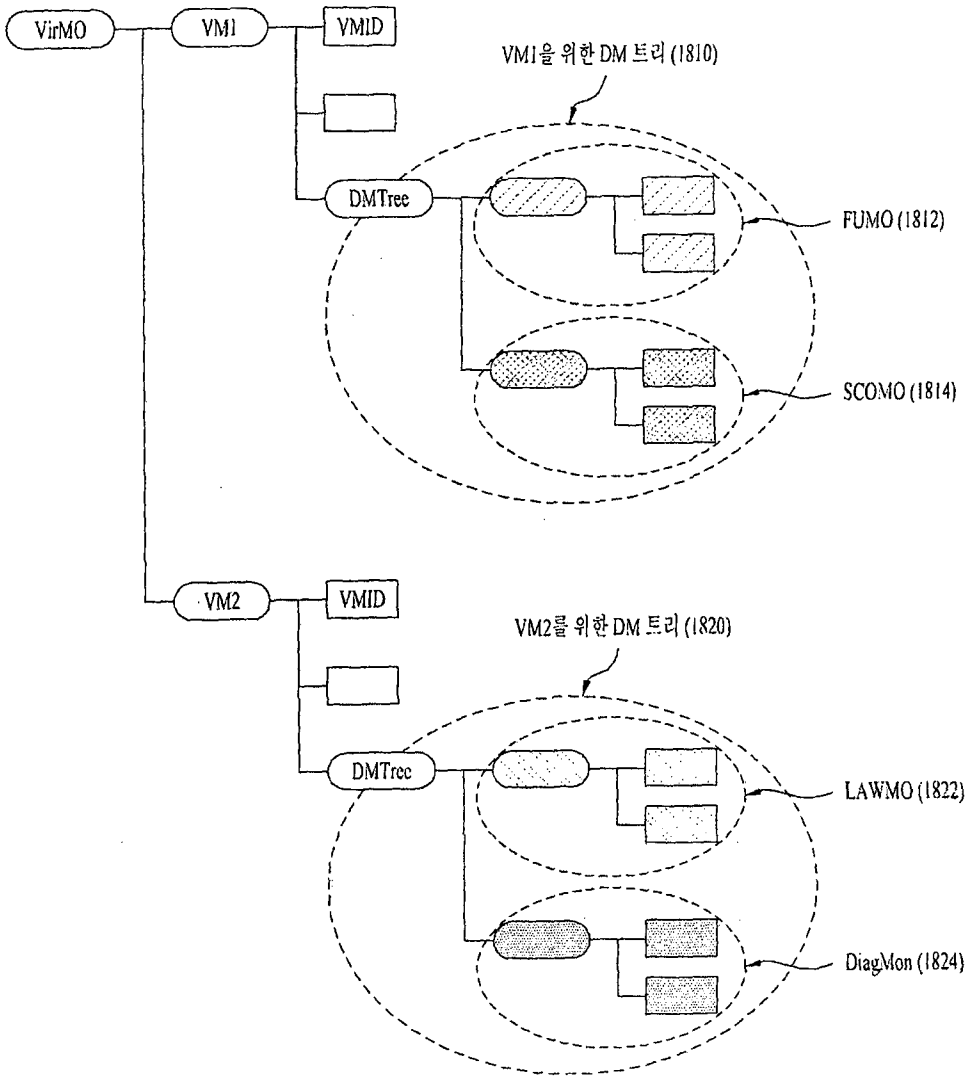
[도 16]


1620
332
334
1630
336
관리 도메인
디바이스 도메인
1610
관리 VM
엔터프라이즈 VM
개인용 VM
GwMO 서버
GwMO 콤포넌트
GwMO 클라이언트
240
DM 서버
DM 클라이언트
DM 서버
DM 클라이언트
210
1640
320
하이퍼바이저
310
하드웨어
1230

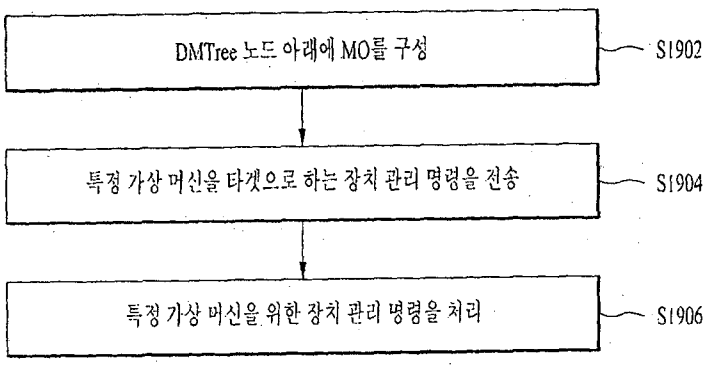
[도 17]


334
엔터프라이즈 VM
MO 클라이언트
1540
336
개인용 VM
MO 클라이언트
1520
332
관리 VM
VirMO 클라이언트
DM 클라이언트
220
210
하이퍼바이저
320
하드웨어
310
250
VirMO 서버
DM 서버
240

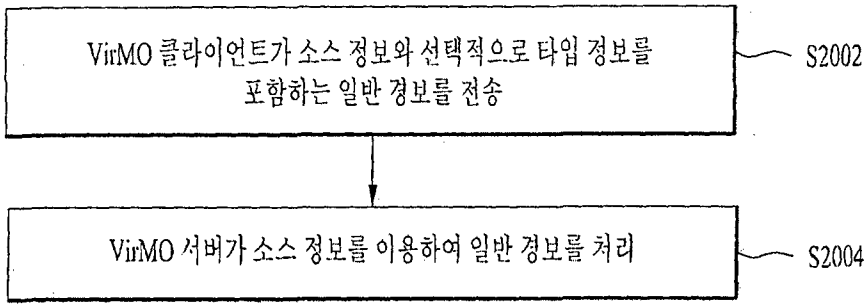
[도 18]


VirMO
VM1
VMID
VM1을 위한 DM 트리 (1810)
DMTree
FUMO (1812)
SCOMO (1814)
VM2
VMID
VM2를 위한 DM 트리 (1820)
DMTree
LAWMO (1822)
DiagMon (1824)

[도 19]


DMTree 노드 아래에 MO를 구성
S1902
특정 가상 머신을 타겟으로 하는 장치 관리 명령을 전송
S1904
특정 가상 머신을 위한 장치 관리 명령을 처리
S1906

[도 20]



[도 21]

```
<Alert>
  <CmdID>2</CmdID>
  <Data>1226</Data>    <!-- Generic Alert -->
  <Item>
    <Source><LocURI>./VirMO/VM1/DMTree/FUMO</LocURI></Source>
    <Meta>
      <Type xmlns='syncml:metinf'>
                org.openmobilealliance.dm.firmwareupdate.devicerequest
      </Type>
      <Format xmlns='syncml:metinf'>xml</Format>
    </Meta>
    <Data>
      <!-- Data Goes Here -->
    </Data>
  </Item>
</Alert>
```

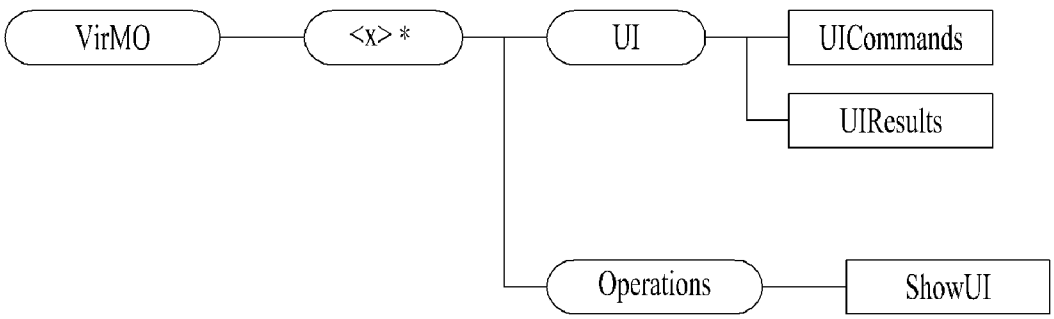
[Fig. 22]

```
<Alert>
  <CmdID>2</CmdID>
  <Data>1226</Data>    <!-- Generic Alert -->
  <Item>
    <Source><LocURI>./VirMO/VM1/DMTree/FUMO</LocURI></Source>
    <Meta>
      <Type xmlns='syncml:metinf'>
            org.openmobilealliance.dm.firmwareupdate.devicerequest-vm
      </Type>
      <Format xmlns='syncml:metinf'>xml</Format>
    </Meta>
    <Data>
      <!-- Data Goes Here -->
    </Data>
  </Item>
</Alert>
```

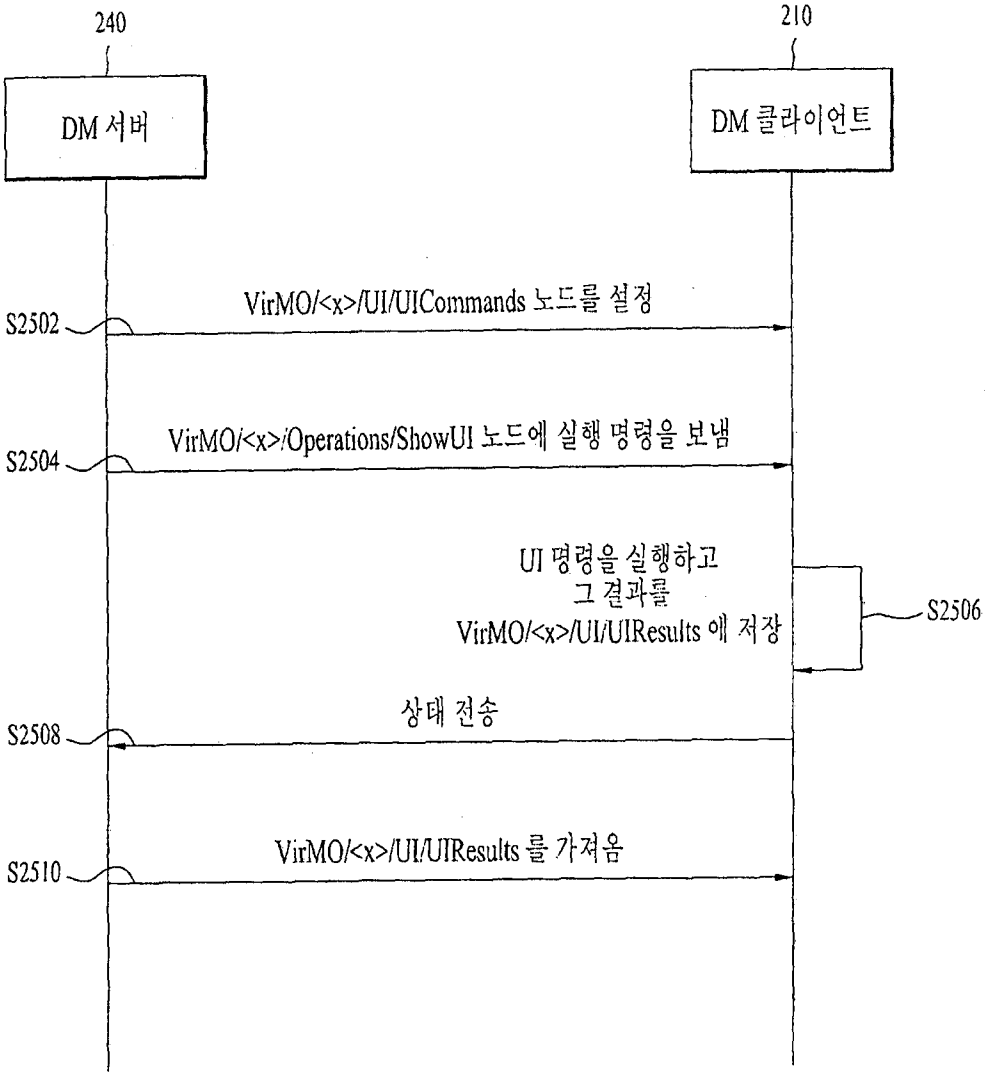
[Fig. 23]

```
<Alert>
  <CmdID>2</CmdID>
  <Data>1100</Data>
  <Item><Data>MINDT=10</Data></Item>
  <Item>
    <Data>Management in progress</Data>
  </Item>
</Alert>
```

[Fig. 24]



[도 25]


240
210
DM 서버
DM 클라이언트
S2502
VirMO/<x>/UI/UICommands 노드를 설정
S2504
VirMO/<x>/Operations/ShowUI 노드에 실행 명령을 보냄
UI 명령을 실행하고
그 결과를
VirMO/<x>/UI/UIResults 에 저장
S2506
S2508
상태 전송
S2510
VirMO/<x>/UI/UIResults 를 가져옴

[도 26]

```
<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>
```

(a)

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

(b)

[도 27]