(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2013년12월02일
(11) 등록번호 10-1334938
(24) 등록일자 2013년11월25일

(51) 국제특허분류(Int. Cl.)
G06K 17/00 (2006.01) *G06F 9/44* (2006.01)
G06Q 20/04 (2012.01)
(21) 출원번호 10-2012-0066384
(22) 출원일자 2012년06월20일
심사청구일자 2012년06월20일
(56) 선행기술조사문헌
KR1020090072623 A*
KR1020110073626 A*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
주식회사 한국스마트카드
서울 중구 남대문로5가 581
(72) 발명자
장병근
경기도 용인시 수지구 죽전동 건영캐스빌 504-703
남현우
서울특별시 도봉구 도봉동 579-2
(74) 대리인
양기혁, 김남식, 이인행, 한윤호

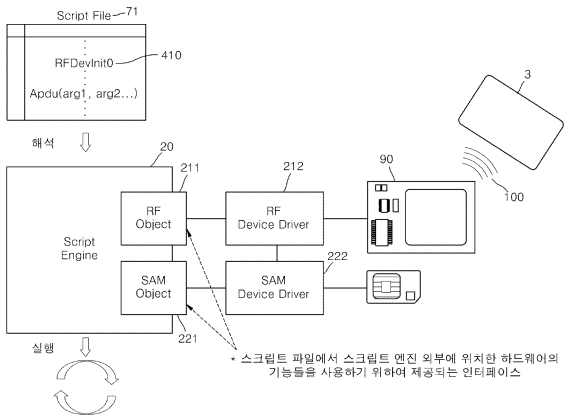
전체 청구항 수 : 총 8 항 심사관 : 홍기완

(54) 발명의 명칭 **스크립트 파일 기반으로 카드 처리를 수행하는 RF 결재 단말기**

(57) 요 약

카드처리 오브젝트를 제공하는 스크립트 엔진을 포함하는 단말기를 제어하는 방법이 공개된다. 이 방법은, 단말기가, 카드처리 오브젝트를 호출하는 코드를 포함하는 스크립트를 로드하는 단계, 및 스크립트 엔진을 이용하여 스크립트를 실행함으로써 단말기의 디바이스 드라이버를 제어하는 단계를 포함한다. 이때, 카드처리 오브젝트는 디바이스 드라이버를 제어하기 위한 함수를 포함할 수 있다.

대 표 도



이 발명을 지원한 국가연구개발사업

과제고유번호	10038474
부처명	지식경제부
연구사업명	산업융합원천기술개발사업(World Best Software)
연구과제명	AFC 표준 SW 솔루션 개발
기 여 율	1/1
주관기관	주식회사 한국스마트카드
연구기간	2010.10.01 ~ 2013.03.31

특허청구의 범위

청구항 1

컴파일 언어에 의해 제어되는 디바이스 드라이버와 상기 컴파일 언어에 의해 생성된 카드처리 오브젝트를 제공하는 스크립트 엔진을 포함하는 단말기에서, 상기 디바이스 드라이버를 제어하는 제어방법으로서,

상기 단말기에서, 상기 카드처리 오브젝트를 호출하는 코드를 포함하는 스크립트를 로드하는 단계; 및

상기 단말기에서, 상기 스크립트 엔진을 이용하여 상기 스크립트를 실행함으로써 상기 디바이스 드라이버를 제어하는 단계를 포함하며,

상기 카드처리 오브젝트는 상기 디바이스 드라이버를 제어하기 위한 함수를 포함하는,

디바이스 드라이버 제어방법.

청구항 2

제1항에 있어서, 상기 함수의 인자에는 수행하고자 하는 특정 스크립트의 파일명이 포함되는 것을 특징으로 하는, 디바이스 드라이버 제어방법.

청구항 3

삭제

청구항 4

제1항에 있어서, 상기 카드처리 오브젝트는 RF 오브젝트 또는 SAM 오브젝트인 것을 특징으로 하는, 디바이스 드라이버 제어방법.

청구항 5

제1항에 있어서, 상기 로드하는 단계 이전에, 상기 단말기를 초기화하는 단계; 및 서버로부터 상기 스크립트를 제공받는 단계를 더 포함하는, 디바이스 드라이버 제어방법.

청구항 6

제1항에 있어서, 상기 스크립트는, 상기 단말기에 저장되어 있거나 서버에 의해 제공되는 복수 개의 스크립트 중 상기 단말기가 수행할 어플리케이션에 따라 선택된 것인, 디바이스 드라이버 제어방법.

청구항 7

제6항에 있어서, 상기 스크립트의 실행결과를 상기 어플리케이션에 리턴하는 단계를 더 포함하는, 디바이스 드라이버 제어방법.

청구항 8

컴파일 언어에 의해 생성된 카드처리 오브젝트를 제공하는 스크립트 엔진;

컴파일 언어에 의해 제어되는 디바이스 드라이버;

메모리; 및

처리부를 포함하며,

상기 처리부는, 상기 카드처리 오브젝트를 호출하는 코드를 포함하는 스크립트를 상기 메모리에 로드하도록 되어 있고, 상기 스크립트 엔진을 이용하여 상기 스크립트를 실행함으로서 상기 디바이스 드라이버를 제어하도록 되어 있으며,

상기 카드처리 오브젝트는 상기 디바이스 드라이버를 제어하기 위한 함수를 포함하는,

단말기.

청구항 9

제8항에 있어서, 상기 함수의 인자에는 수행하고자 하는 특정 스크립트의 파일명이 포함되는 것을 특징으로 하는, 단말기.

청구항 10

삭제

명 세 서

기 술 분 야

[0001] 본 발명은 RF 결재 단말기를 위한 기술에 관한 것으로서, 특히 스크립트를 이용하는 기술에 관한 것이다.

배 경 기 술

[0002] RF 결재 단말기(이하, 간단히 '단말기')는 다양한 상품을 적재한 다양한 종류의 RF 카드 제품의 메모리를 액세스하여 해당 상품에 관한 처리를 수행하도록 하는 실행파일을 포함한다. 또한, 단말기는 카드 유효성 체크, 보안관련 모듈, 및 통신 인터페이스 등 카드를 이용한 전자결재를 지원하기 위한 다양한 어플리케이션(application) 기능 모듈들을 포함할 수 있다. 이때, 새로운 카드 제품이나 상품이 추가되는 경우, 단말기가 이를 처리하기 위한 새로운 어플리케이션 기능을 추가적으로 실행할 수 있도록 단말기의 프로그램을 변경할 필요가 있다.

[0003] 단말기에 포함된 다양한 어플리케이션의 기능들은 단말기의 하드웨어 또는 운영체제에 따라 구성에 차이가 발생할 수 있으며, 성능 이슈로 인하여 현재 C, C++와 같은 로우레벨(low level) 언어로 작성되고 있다. 이와 같은 언어로 작성된 프로그램은 컴파일(compile) 과정을 거쳐 생성된 정적인 실행파일 형태로 구현되고 있다. 단말기의 기능을 업데이트(update)하고자 할 때에, 어플리케이션의 소스코드(source code)를 수정하여 빌드(build)를 다시 수행한다. 컴파일 결과 생성된 실행파일을 단말기에 복사한 후 수행 중인 어플리케이션의 실행 파일을 종료하고 위의 복사된 소프트웨어 실행파일을 재실행함으로써 단말기 기능의 업데이트 적용을 완료할 수 있다. 따라서 단말기는 어플리케이션의 설치 및 재실행 과정 도중에는 운영 중지상태가 된다.

[0004] 카드 처리 단말기는 지불 단말기, 충전 단말기, 및 조회 단말기 등으로 나뉠 수 있다. 예를 들어 지불 단말기의 경우 버스, 택시, 및/또는 지하철 단말기, 그리고 유통 단말기와 같이 다양한 형태로 개발되었다. 현재까지는 각 단말기들의 어플리케이션 개발 시, 단말기의 종류나 제조회사에 따라 동일한 카드 처리 기능들에 대해서도 중복적으로 개발하였으며, 그 결과 동일한 기능들이 각 단말기를 위하여 서로 다른 방식으로 구현되고 최적화 과정을 거치게 되었다. 예를 들어 동일한 지불 기능을 수행하는 지불 단말기들을 제조 회사나 단말기 타입에 따라 개별적으로 개발하였고, 따라서 똑같은 상품을 탑재한 동일 회사의 카드의 지불을 처리하는 경우에도 다양한 처리 소스코드가 발생되었다. 이와 같은 경우 동일한 기능을 수행하는 다양한 코드가 발생되므로 품질 관리가 어려워지며, 중복된 노력과 비용이 발생하였다.

발명의 내용

해결하려는 과제

[0005] 기존에는 단말기의 기능을 추가하거나 변경하기 위해서 소스코드를 수정한 후 컴파일 과정을 수행하였다. 이를 위하여, 기능의 추가나 변경이 발생할 때마다 새로운 실행파일이 빌드 되었다. 따라서 일부 기능을 업데이트하기 위해 전체 어플리케이션이 새로 컴파일 되어야 했다. 즉, 일부 기능만을 변경한다고 하더라도 전체 어플리케이션에 영향을 미치게 되어 유연한 시스템의 구성이 어려웠다.

[0006] 본 발명의 목적은 RF 결재 단말기의 어플리케이션의 기능 추가 및 변경 등의 작업을 소스 코드의 컴파일 과정과 설치 후 재실행과 같은 단계 없이, 스크립트 파일 다운로드만으로 동적으로 기능의 추가 및 변경이 가능한 단말기 기술을 제공하는 것이다.

과제의 해결 수단

[0007] 본 발명에서는 단말기 어플리케이션의 기능 중 C 또는 C++로 작성된 카드처리 모듈의 기능들을 스크립트 파일을

사용하여 처리할 수 있는 시스템의 구조 및 처리 방법을 제공한다.

[0008] 이를 위해, 단말기의 RF 카드 처리를 통한 지불, 충전, 거래내역 조회, 잔액조회, 및 서비스 변경 등의 기능들을 스크립트 형태로 모듈화 하여 제공한다. 그 결과 어플리케이션을 수정하지 않더라도, 원하는 기능의 스크립트를 적용함으로써 다양한 카드 처리 기능들을 사용할 수 있다.

[0009] 따라서 단말기는, 예컨대, 지불, 충전, 및 조회의 기능 중 어느 하나의 기능에 종속되지 않으며, 탑재되는 스크립트에 따라 어느 하나 이상의 기능을 자유롭게 수행할 수 있다. 또한 단말기 기능의 업데이트를 위하여 버전업(version-up)을 수행하거나 패치(patch)가 필요할 때에도, 스크립트 교체만으로 시스템 중단 없이 단말기의 기능을 추가하거나 변경할 수 있는 유연성을 갖출 수 있다.

[0010] 이를 위하여, 본 발명은 어플리케이션의 시스템 구성과 스크립트 처리 방법에 대해 제안하고 있다. 특히 RF 카드의 처리 기능이 탑재된 결재 단말기에서 지원 가능한 카드의 종류나 상품이 추가되었을 때에, 업데이트 된 스크립트를 교체하는 것만으로 업데이트가 완료된다.

[0011] 본 발명의 일 양상에 따르면 카드처리 오브젝트를 제공하는 스크립트 엔진을 포함하는 단말기를 제어하는 방법을 제공할 수 있다. 이 방법은, 단말기에서, 위의 카드처리 오브젝트를 호출하는 코드를 포함하는 스크립트를 로드하는 단계, 및 위의 단말기에서, 위의 스크립트 엔진을 이용하여 위의 스크립트를 실행함으로써 위의 단말기의 디바이스 드라이버를 제어하는 단계를 포함한다.

[0012] 이때, 위의 카드처리 오브젝트는 위의 디바이스 드라이버를 제어하기 위한 함수를 포함할 수 있다.

[0013] 이때, 위의 함수는 컴파일 언어로 기술되어 있을 수 있다.

[0014] 이때, 위의 카드처리 오브젝트는 RF 오브젝트 또는 SAM 오브젝트인 것을 특징으로 할 수 있다.

[0015] 이때, 위의 단말기 제어 방법은, 위의 로드하는 단계 이전에, 위의 단말기를 초기화하는 단계, 및 서버로부터 위의 스크립트를 제공받는 단계를 더 포함할 수 있다.

[0016] 이때, 위의 스크립트는, 위의 단말기에 저장되어 있거나 서버에 의해 제공되는 복수 개의 스크립트 중 위의 단말기가 수행할 어플리케이션에 따라 선택된 것일 수 있다.

[0017] 이때, 위의 스크립트의 실행결과를 위의 어플리케이션에 리턴(return)하는 단계를 더 포함할 수 있다.

[0018] 본 발명의 다른 양상에 따라, 스크립트를 이용하는 단말기를 제공할 수 있다. 이 단말기는, 카드처리 오브젝트를 제공하는 스크립트 엔진, 위의 단말기의 디바이스를 위한 디바이스 드라이버, 메모리, 및 처리부를 포함한다. 이때, 위의 처리부는, 위의 카드처리 오브젝트를 호출하는 코드를 포함하는 스크립트를 위의 메모리에 로드하도록 되어 있고, 위의 스크립트 엔진을 이용하여 위의 스크립트를 실행함으로서 위의 디바이스 드라이버를 제어하도록 되어 있다.

[0019] 이때, 위의 카드처리 오브젝트는 위의 디바이스 드라이버를 제어하기 위한 함수를 포함할 수 있다. 이때, 위의 함수는 컴파일 언어로 기술되어 있을 수 있다.

발명의 효과

[0020] 본 발명의 일 실시예에 따르면, 단말기에서는 특정 기능의 처리 코드가 명시된 스크립트 파일을 다운로드(download)한 후에 내부 메모리에 로드(load)하여 준비를 하고, 그 다음 해당 기능이 요청된 시점에 그 스크립트 파일을 실행하게 된다. 따라서 기존 단말기에서와 같이 어플리케이션의 기능 수정을 위해 소스 코드를 수정한 후 컴파일 하여 실행 바이너리(binary)를 단말기에 설치하는 일련의 과정들이 필요 없어진다. 그 결과, 컴파일 과정이 필요 없어지게 되므로 단말기가 동작하는 도중에도 동적으로 스크립트를 교체함으로써 원하는 기능을 수행할 수 있게 된다. 그리고 어플리케이션의 변경을 하지 않고, 단지 해당된 스크립트만 교체하게 되면 기능 변경이 완료될 수 있다. 따라서 기존 시스템에서는 불가능했던 시스템의 무중단 상태에서도 업데이트가 가능해졌으며, 기존 대비 업데이트에 소요되는 노력과 시간을 절감 시킨다.

[0021] 본 발명의 일 실시예에 따르면, 지불 및 충전과 같은 카드 처리를 위한 기능들을 스크립트 파일로 작성한 후 이를 수행하기 위한 스크립트 엔진과 관련 디바이스 인터페이스를 포함하고 있는 임의의 단말기에 제공해 주기 때문에, 이 임의의 단말기가 동일한 하나의 스크립트 파일로 해당 기능들을 수행할 수 있게 된다.

[0022] 또한, 본 발명의 일 실시예에 따르면, 스크립트 엔진과 카드 처리 관련 모듈을 이용함으로써 범용 단말기가 지불 또는 충전과 같은 다양한 기능을 수행할 수 있도록 할 수 있다. 따라서 본 발명에 따르면 단말기를 특정 목

적으로만 사용할 수도 있고, 또는 PC와 같이 범용으로 사용할 수도 있다. 예컨대, 단말기를 범용으로 제작하되 RF 태그 모듈과 SAM 모듈을 포함하게 되면, 이 모듈을 구동하는데 필요한 스크립트만을 서버에서 다운로드 받아 다양한 기능을 실행할 수 있다.

[0023] 본 발명의 일 실시예에 따르면, 새로운 카드 타입이나 카드 상품이 출시된다고 하더라도, 해당 카드나 상품을 처리할 수 있는 스크립트 파일만을 다운로드 받아 교체만 해주면 되기 때문에 유연하게 시스템의 정지 없이도 업데이트가 가능하다.

[0024] 또한 전용으로 개발된 어플리케이션만 실행할 수 있었던 기존 결재 단말기와 달리, 스크립트 파일을 해석하고 수행할 수 있는 스크립트 엔진을 이용함으로써, 단말기에 탑재된 스크립트의 종류에 따라 단말기를 지불 단말기, 충전 단말기 등의 다양한 형태로 변경하여 사용할 수 있다.

도면의 간단한 설명

[0025] 도 1은 본 발명의 일 실시예에 따른 단말기의 구성도를 설명하기 위한 것이다.

도 2는 본 발명의 일 실시예에 따른 스크립트 기반의 단말기에서의 카드 처리 프로세스를 나타낸다.

도 3은 본 발명의 일 실시예에 따른 스크립트 모듈의 상세 처리 단계를 나타낸 것이다.

도 4는 본 발명의 일 실시예에 따른 단말기에 포함되는 스크립트 파일, RF 오브젝트, 및 SAM 오브젝트들 간의 관계를 예시한 것이다.

도 5는 APDU 명령어의 구조를 일부 나타낸 것이다.

발명을 실시하기 위한 구체적인 내용

[0026] 이하, 첨부한 도면을 참고로 하여 본 발명의 실시예에 대하여 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자가 용이하게 실시할 수 있도록 상세히 설명한다. 그러나 본 발명은 여러 가지 상이한 형태로 구현될 수 있으며 여기에서 설명하는 실시예에 한정되지 않는다. 이하에서 사용되는 용어는 단지 특정 실시예를 언급하기 위한 것이며, 본 발명을 한정하는 것을 의도하지 않는다.

[0027] 도 1은 본 발명의 일 실시예에 따른 단말기의 구성도를 설명하기 위한 것이다.

[0028] 도 1에 나타낸 단말기(1)는 카드 처리부(10), 스크립트 엔진(20), 디바이스 제어부(30), 통신 인터페이스(40), 및 사용자 인터페이스(User Interface)(50)를 포함할 수 있다. 이때, 카드 처리부(10)는 RF(Radio Frequency, 무선) 방식에 의하여 액세스되는 외부의 전자결제 카드와 관련된 지불 기능 및 충전 기능 등을 처리할 수 있도록 되어 있다. 그리고 디바이스 제어부(30)는 RF 디바이스 제어부 및 SAM 디바이스 제어부를 포함할 수 있다. 여기서, 사용자 인터페이스(50)는 사용자에게 화면 디자인을 보여주고, 사용자로부터 입력을 받도록 되어 있을 수 있다.

[0029] 또한, 단말기(1)는 카드 처리를 위한 카드 유효성 체크부(61), 보안 및 무결성 처리부(62), 버스 컴포넌트 처리부(63), 철도 컴포넌트 처리부(64), 및 요금제 처리부(65)와 같은 기능 모듈들을 포함할 수 있다.

[0030] 서버(2)는 전자결제 카드와 관련된 지불, 충전, 거래내역확인, 잔액확인, 및 서비스 갱신 등과 같은 기능을 처리하기 위한 스크립트 파일들(71, 711~715)을 포함하는 스크립트 DB(데이터베이스)(70)에 액세스할 수 있도록 되어 있다. 또한, 서버(2)는 스크립트 DB(70)에 포함된 임의의 스크립트를 단말기(1)의 카드 처리부(10)를 통해 단말기(1)에게 전송할 수 있다. 스크립트 파일들(711~715)은 서버(2)를 이용하지 않고, 별도의 USB와 같은 저장장치를 이용하여 단말기(1)에 전달될 수도 있다.

[0031] 임베디드 시스템의 경우 성능 이슈 때문에 C, C++과 같은 로우레벨 언어로 작성된다. 그런데 본 발명의 일 실시예처럼, ISO-7816 또는 ISO-14443와 같은 국제 표준을 따르는 카드처리의 경우, 동일한 카드상품이라면 공통된 APDU(Application Protocol Data Unit) 명령어로 처리가 가능하다. 그리고 카드처리 기능의 경우 단말기의 하드웨어나 운영체제에 따른 차이가 적으며, 처리속도 측면에서도 명령어 처리에 소요되는 시간보다는 RF 디바이스 및 SAM 디바이스 간의 통신에서 발생되는 처리시간이 큰 비중을 차지한다. 따라서 카드처리 기능을 스크립트 방식으로 구현하는 경우에도 C 또는 C++로 구현하는 경우와 비교하여 충분한 속도를 보장할 수 있다.

[0032] 도 1과 같은 스크립트 파일 기반의 결재 단말기 시스템에서는 지불 기능이 추가적으로 수정되거나 버전(version) 증가에 따른 업데이트가 필요한 경우에, 단지 스크립트 파일을, 예컨대 서버로부터 다운로드하여 교

체하기만 하면 업데이트를 완료할 수 있다. 이와 마찬가지로, 기능을 추가하는 경우에도 해당 추가 기능을 위한 스크립트 파일을 서버로부터 다운로드하여 해당 추가 기능이 필요한 시점에 스크립트 파일을 수행하면 그 기능을 수행할 수 있다.

[0033] 이때, 스크립트 언어로서, 신규 언어를 개발하고 이를 해결할 수 있는 인터프리터를 개발하여 사용할 수 있다. 또는 도 1의 스크립트 엔진(20)을 이용할 수 있다. 스크립트 엔진(20)으로는 Spider Monkey 또는 V8과 같은 것을 사용할 수 있으며, 이때 자바 스크립트와 같이 표준적인 스크립트 언어로 작성된 프로그램을 이 스크립트 엔진을 기반으로 수행할 수 있다.

[0034] 단말기(1)에서 수행되는 어플리케이션 중 대표적인 카드처리 기능인 지불, 충전, 거래내역확인, 잔액확인, 및 서비스 갱신 등의 기능을 스크립트 언어로 작성할 수 있다. 이 스크립트 파일은 처음 설치 시 단말기(1)에 저장되거나 서버(2)로부터 다운로드 받아 단말기(1)에 저장할 수 있다.

[0035] 스크립트 엔진(20)에서는 카드 처리를 위한 RF 오브젝트와 카드 보안 처리를 위한 SAM 오브젝트가 제공될 수 있다. 스크립트 파일(71)은 이와 같은 오브젝트들을 이용하여 교통 카드의 지불 및 충전 등의 카드처리 작업을 수행할 수 있다

[0036] 도 2는 본 발명의 일 실시예에 따른 스크립트 기반의 단말기에서의 카드 처리 프로세스를 나타낸다.

[0037] 단계(S21)에서 단말기는 처음 부팅이 된 후 단말기 초기화 과정을 거쳐 카드 처리에 필요한 스크립트 파일(71)을 로드한다. 만약 단말기(1)가 지불 단말기라면 지불 기능을 수행하는 지불기능-스크립트 파일(711)을 로드하고, 충전 단말기라면 충전 기능을 수행하는 충전기능-스크립트 파일(712)을 로드할 수 있다. 단지 조회 기능만을 수행하도록 되어 있는 단말기라면 잔액확인 기능 또는 거래내역확인 기능을 수행하는 확인-스크립트 파일(713)을 로드할 수 있다. 이렇게 기능별 스크립트 파일들이 로드된 이후에, 카드 태그가 발생하면 해당 기능에 매칭되는 스크립트 파일(71)이 호출되어 처리된다. 그 결과는 어플리케이션에 리턴(return)될 수 있다. 처리과정을 자세히 살펴보면 카드태그 대기 상태에서 RF 디바이스에 카드가 인접하여 인식되면 카드의 존재가 검출(detect)되고, 그 다음 카드의 데이터를 읽는 리드(read) 과정이 수행될 수 있다.

[0038] 단계(S22)에서는, 위와 같이 리드된 카드의 데이터를 기반으로 단말기의 목적(지불, 충전, 또는 내역 조회)에 해당하는 스크립트 파일(71)에 대한 수행 요청을 하게 된다. 스크립트 파일(71)은 단말기 초기 부팅 시 로드될 수 있으며, 단말기가 동작하는 도중에도 로드될 수 있다. 이때, 로드되는 스크립트 파일(71)은, 예컨대 서버(2)로부터 다운받을 수도 있다.

[0039] 단계(S23)에서는, 해당하는 스크립트 파일(71)이 수행된다. 예컨대, 지불 기능의 수행이 요구될 때에는 지불 스크립트를 수행하고(단계 S231), 충전 기능의 수행이 요구될 때에는 충전 스크립트를 수행하고(단계 S232), 또는 잔액확인 기능의 수행이 요구될 때에는 잔액확인 스크립트를 수행할 수 있다(단계 S233). 이때 단계(S23)를 수행할 때에, 각 기능을 위한 각각의 스크립트 파일에서는 RF 무선 상태에서 카드와 통신을 위해 RF 오브젝트에게 APDU 명령 요청을 하여 RF 통신 기능을 수행할 수 있다(단계 S241). 이와 마찬가지로 단계(S242)에서는, SAM 오브젝트가 스크립트 파일(71)에서 요청하는 SAM 통신 기능을 제공할 수 있다. 각각의 스크립트 파일(71)은 스크립트 엔진(20) 상에서 수행될 수 있다. 단계(S25)에서는, 스크립트의 수행이 완료된 결과 발생하는 처리값이 어플리케이션에 리턴될 수 있다.

[0040] 도 3은 본 발명의 일 실시예에 따른 스크립트 모듈의 상세 처리 단계를 나타낸 것이다.

[0041] 도 2에서 이미 설명한 바와 같이, 단말기(1)는 부팅 시 단말기(1)를 초기화 한 후, 지원하고자 하는 기능의 스크립트 파일(71)을 메모리에 로드한다. 스크립트 파일(71)들은 미리 디스크(저장부, 80)에 저장되어 있거나 서버(2)로부터 다운로드 받을 수 있다. 여러 종류의 스크립트 파일(71)이 존재할 수 있다.

[0042] 도 3을 참조하면, 단계(S31)에서, 단말기(1)의 카드처리부(10)는, 단말기(1)가 부팅될 때에, 필요한 기능에 해당하는 스크립트 파일을 메모리(81) 또는 스크립트 엔진(20)에 등록(로드)하도록 요청할 수 있다. 이때, 스크립트 파일들(711, 712, 713)은 단말기(1)의 저장부(80) 저장되어 있을 수 있으며, 이 저장된 스크립트 파일들(711, 712, 713)은 서버(2)로부터 제공받은 것일 수도 있다.

[0043] 그 다음, 단계(S32)에서, 해당되는 스크립트 파일(711, 712, 713)들을 검색하여 메모리(81) 또는 스크립트 엔진(20)에 로드(등록)한다. 로드가 끝나면, 단말기 시스템에서는 카드에 대한 처리가 준비된 상태가 된다.

[0044] 이때, 실제 카드가 RF 안테나에 인접하여 태그 되어 카드가 검출되게 되면(S33), 리드 과정(S34)을 거쳐, 스크

립트 수행을 요청한다(S35).

[0045] 스크립트 엔진(20)에서는 스크립트 파일 처리에 필요한 카드(card) 오브젝트, 예컨대 RF 오브젝트와 SAM 오브젝트를 제공할 수 있다. 또한 스크립트 엔진(20)으로서 Rhino, Spider Monkey, V8 등의 범용 자바 스크립트 엔진을 사용할 수 있다. 어떤 스크립트 파일(71)을 호출할 것인지는 어플리케이션에서 처리해야 할 기능들을 판단한 후 선택될 수 있다.

[0046] 단계(S36)에서는, 메모리에 로드된 한 개 이상의 스크립트 파일(71)을 이용하여, 한 번에 하나 또는 여러 개의 스크립트 파일(71)을 한 번에 수행할 수 있다. 즉 지불 스크립트(711)를 수행하면서 잔액확인 스크립트(713)를 수행할 수도 있으며, 만약 충전 수수료를 받는 비즈니스가 있다고 한다면 충전 스크립트(712) 수행 후 수수료 결재를 위한 지불 스크립트(711)를 수행할 수 있다.

[0047] 단계(S37)에서는, 스크립트의 수행이 완료된 결과 발생하는 처리 결과가 저장될 수 있다.

[0048] 도 4는 본 발명의 일 실시예에 따른 단말기에 포함되는 스크립트 파일, RF 오브젝트, 및 SAM 오브젝트들 간의 관계를 예시한 것이다.

[0049] 스크립트 파일(71)에는 특정 기능들을 수행하기 위하여 작성된 동작들을 나타내는 코드(410)가 명시되어 있다. 스크립트 파일(71)이 단말기에서 수행되기 위해서는 스크립트 엔진(20)을 통해야 한다. RF 지불결재 단말기에서는 비연결 상태에서 카드처리를 위하여 RF 디바이스를 제어할 수 있어야 하며, 보안 기능들을 수행하기 위하여 SAM 디바이스를 제어할 수 있어야 한다.

[0050] 그러나 스크립트 파일(71)로는 이와 같은 하드웨어들을 직접 제어할 수 없다. 따라서 기존에 C, C++ 등으로 작성된 하드웨어 제어 코드들을 사용할 수 있다. 따라서 스크립트 엔진(20) 외부에 위치한 RF 디바이스 드라이버 및/또는 SAM 디바이스 드라이버를 제어할 수 있도록, 스크립트 엔진(20)에서는 RF 오브젝트(211) 및/또는 SAM 오브젝트(221)를 제공할 수 있다. 즉, 위와 같은 RF/SAM 오브젝트들(211, 221)은 스크립트 파일(71)에서 스크립트 엔진(20)의 외부에 위치한 하드웨어의 기능들을 사용하기 위하여 제공되는 인터페이스로 간주할 수 있다.

[0051] 스크립트 파일(71)은 스크립트 엔진(20)이 제공하는 오브젝트를 이용하여 하드웨어를 제어할 수 있고, 따라서 카드를 위한 기능을 처리할 수 있다. 예컨대, 도 4와 같이 RF 오브젝트(211)가 RF 디바이스 드라이버(212)를 통해 RF 하드웨어(90)를 제어함으로써, 카드(3)에게 RF 통신을 통해 APDU 명령어(100)를 전달하고 이에 대응하는 응답 처리를 수행할 수 있다.

[0052] APDU 명령어(100)는 카드(3)와의 통신을 위해 명령어 전달 표준이다. APDU 명령어(100)는 크게 두 가지로 나뉜다. 단말기(1)에서 카드(3)의 명령어를 수행하기 위해서 전송하는 '명령 APDU'와, 카드(3)에서 명령어 처리가 완료된 후 카드(3)에서 단말기(1)로 전송되는 '응답 APDU'가 있다. 이와 같은 APDU 명령어 프로토콜은 ISO 7816-4 국제표준에서 규정하고 있다. 단말기에서는 APDU 프로토콜을 사용하여 카드에서 제공하는 명령어들을 수행하여 카드 처리를 수행하고 있으며, 이를 통해 지불, 충전, 및 잔액확인과 같은 다양한 RF 결재 단말기 기능들이 수행된다. 실제 스크립트 파일에서는 이와 같은 APDU 명령어들을 실행하여 카드처리를 수행하게 된다.

[0053] 도 5는 APDU 명령어의 구조를 일부 나타낸 것이다.

[0054] '명령 APDU'는 클래스 바이트(command-ID)(CLA), 명령어 코드(INS), 및 파라미터(P1, P2)를 포함하는 명령 APDU 헤더를 포함하며, 명령 APDU 데이터 길이(Lc), 명령 APDU 데이터(Data), 및 응답 APDU 상의 데이터 길이(Le)를 포함한다. '응답 APDU'는, 응답(Data), 및 카드 상태 코드(SW1, SW2)를 포함한다.

[0055] <스크립트 코드의 예>

[0056] <표 1>은 교통카드의 잔액확인을 위하여 자바 스크립트로 작성된 스크립트 파일의 예이다. 스크립트의 내용을 살펴보면 카드 처리를 위해 RF 디바이스를 초기화 하고(Line 3), 이후 잔액확인을 위한 APDU 명령을 수행한다(Line 10). APDU의 처리 후 리턴 받은 SW 변수(sw1, sw2)를 참조하여 처리 결과를 확인하고 있으며(Line 19), 정상적으로 APDU의 리턴 값을 받았다면 리턴 값을 디코딩 하여 잔액 값을 확인할 수 있다(Line 27). <표 1>의 예제 스크립트는 ECMA 표준을 따르는 자바 스크립트로 작성된 것이며, 웹브라우저에서 많이 사용 중인 자바 스크립트 엔진인 Rhino, Spider Monkey, 또는 V8등의 엔진을 사용할 수 있다. 참고로 위 스크립트 파일은 Spider Monkey 엔진에서 문제없이 수행되었다.

[0057] <표 1>은 잔액확인을 위한 간단한 스크립트의 예제이지만, 지불 또는 충전 등의 기능들을 SAM 오브젝트를 추가

사용하여 작성 가능하다.

표 1

[0058]

```
Line 1:: load("tools/wbsutil.js");

Line 2:: // RF 디바이스(RC531)를 초기화
Line 3:: RfDevInit();

Line 4:: var bCLA = 0x90;
Line 5:: var bINS = 0x4C;
Line 6:: var bP1 = 0x00;
Line 7:: var bP2 = 0x00;
Line 8:: var bLc = 0x00;
Line 9:: var bLe = 0x04;

Line 10:: var retVal = lapdu([bCLA, bINS, bP1, bP2, bLc, bLe])
Line 11:: print("[0]" + retVal[0] + ", [1]" + retVal[1] +", [2]" + retVal[2] +", [3]" +
retVal[3] );

Line 12:: // APDU 리턴값
Line 13:: var val1 = retVal[0];
Line 14:: var val2 = retVal[1];
Line 15:: var val3 = retVal[2];
Line 16:: var val4 = retVal[3];
Line 17:: var sw1 = retVal[4];
Line 18:: var sw2 = retVal[5];

Line 19:: print("SW[0]:" + sw1 + ", SW[1]:" + sw2);

Line 20:: if(sw1 == 0 && sw2 == 0)
Line 21:: {
Line 22::       print("APDU 리턴값 없음");
Line 23:: }
Line 24:: else
Line 25:: {
Line 26::       sum = (val1 << 24) + (val2 << 16) + (val3 << 8) + val4;
Line 27::       print("신카드 잔액 : " + sum);
Line 28:: }
```

[0059] <표 1>에서 RfDevInit( )와 lapdu( )는 RF 디바이스를 호출하는 함수이다. RfDevInit( )는 RF 장치 초기화 함수이며 lapdu( )는 APDU 명령어를 전달하고 리턴값을 전달받는 함수이다. 특히, RfDevInit( )는 RF 디바이스(예: RC531 모듈)를 초기화 해주는 함수로, 초기화가 완료되면 비접촉으로 단말기(1)와 카드(3)가 통신을 수행할 수 있다. 실질적으로 자바 스크립트(71)에서 해당 초기화 함수 RfDevInit( )를 호출하면 C 코드로 작성된 초기화 함수를 호출하게 되며, 이 초기화 함수의 소스코드는 아래의 <표 2>와 같이 주어질 수 있다.

표 2

[0060]

```
119 static JSBool
120 _RfDevInit(JSContext *cx, JSObject *obj, uintN argc, jsval *argv, jsval *rval)
121 {
122 int rc;
123 int iResult;
124 int sresult;
125
126 printf("_RfDevInit() called...□");
127
128 T_CARD_BASIC_INFO stCardInfo;
129
130 iResult = TRfDevInit(DEV_RC531);
131 if (iResult == ERR_DEVICE_RF_RC531_OPEN)
132 {
133 // 7144, 7145 Event 생성
134 }
135
136 //사용카드 등록..
137 TCardAddRfType(CARD_RF_TYPE_ALL);
138 TCardAddPlatformType(CARD_PLATFORM_TYPE_ALL);
139
140 // T-money 신선불 등록
141 T_CARD_APP_PROPERTIES stSetCardInfo;
142 TCardScTmGetProperties(&stSetCardInfo);
143 rc = TCardAddAppProperties(&stSetCardInfo);
144 if ( rc < 0 )
145 {
146 TPrintError("rc=%x:%s", rc, TGetErrStr(rc));
147 }
148
149 // 카드 Detect
150 if ((sresult = TCardDetect(&stCardInfo)) == 0)// SUCCESS
151 {
152 printf("□카드인식완료. CSN: %u□",
TUtilLittleEndian2DWord(stCardInfo.abCSN) );
153 }
. . . .
163 return JS_TRUE;
164 }
```

[0061] <표 2>에 따른 코드에 따라, 디바이스를 초기화 하고(Line 130), 사용 카드와 지원 플랫폼 타입을 등록하고(Line 137, 138), 마지막으로 티머니 카드 어플리케이션 타입을 등록(Line 143)함으로써 초기화 과정을 수행한다. 초기화 과정이 끝나면 카드 태그 대기 상태로 카드의 접근을 기다릴 수 있다(Line 150).

[0062] 다음으로 단말기(1)와 카드(3)는 APDU 명령어를 사용하여 통신을 수행하는데, 이를 위해 스크립트 파일(71)에서는 스크립트에서 사용 가능한 API인 lapdu( ) 함수를 사용할 수 있으며, 이때 C 코드에서는 디바이스를 제어하는 sendApduCard( ) 함수가 호출될 수 있다. sendApduCard( )의 소스코드는 아래의 <표 3>의 예와 같다.

표 3

[0063]

```
60 int sendApduCard(APDU_DataType *adApduSend, APDU_DataType *adApduReceive)
61 {
62 int idx;
63 int nResult;
64
65 fprintf(gOutFile, "◘Card adApduSend...◙");
66 for (idx = 0; adApduSend->length--; idx++)
67 {
68 fprintf(gOutFile, " [%d]=%d", idx, adApduSend->cbApduData[idx]);
69 }
70
71
72 // TODO: send & receive native C code hear
73
74 ////////////////////////////////////////////////////////////
75 char abResBuf[MAX_APDU_SIZE];
76 unsigned char bResLen;
77 char abSW[2] = {0, };
78 RF_APDU stAPDU;
79
80 stAPDU.bCLA = adApduSend->cbApduData[0];
81 stAPDU.bINS = adApduSend->cbApduData[1];
82 stAPDU.bP1 = adApduSend->cbApduData[2];
83 stAPDU.bP2 = adApduSend->cbApduData[3];
84 stAPDU.bLc = adApduSend->cbApduData[4];
85 stAPDU.bLe = adApduSend->cbApduData[5];
88
89 // Send APDU command
90 nResult = RfSCSendApdu(&stAPDU, RF_APDU_CASE2,
91 adApduReceive->cbApduData,
92 &bResLen, abSW);
93
94 ////////////////////////////////////////////////////////////
95
96 fprintf(gOutFile, "◘Card ApduReceived◙");
97
98 if (nResult < 0) // error
99 {
100 adApduReceive->length = 0;
101 return nResult;
102 }
103
104
105 // namhw ////////////////////////////////////////////////
106 adApduReceive->cbApduData[bResLen++] = abSW[0];
107 adApduReceive->cbApduData[bResLen++] = abSW[1];
108 adApduReceive->length = bResLen; // SW값을 때문에 크기가 2바이트가 증가됨
109
110 nResult = adApduReceive->length;
111 ////////////////////////////////////////////////////////////
112
113 return nResult; // error NO ~ -x
114 }
```

[0064] <표 3>의 소스코드에서는, 최종적으로 APDU 명령을 전달하기 위하여 RfSCSendApdu( ) 함수를 호출하여 실제 단말기(1)에서 카드(3)로 APDU 명령어(100)를 전달한다(Line 90).

[0065] <표 2>와 <표 3>에서 설명한 두 함수 RfDevInit( )와 lapdu( )와 같은 스크립트에서 사용하는 RF 오브젝트(211)에서 호출되는 함수들, 예컨대 디바이스 드라이버를 초기화하는 TRfDevInit(), 지원하고자 하는 RF의 타입을 등록하는 TCardAddRfType(), APDU 명령어를 전송하는 RfSCSendApdu() 등의 함수는 RF 디바이스 드라이버(212)에서 제공되는 것이다. 즉, RF 오브젝트(211)는 스크립트 파일(71)과 RF 디바이스 드라이버(212) 사이에서 인터페이스 역할을 할 수 있다. 이와 마찬가지로, SAM 오브젝트(221)는 스크립트 파일(71)과 SAM 디바이스 드라이버(222) 사이에서 인터페이스 역할을 할 수 있다.

[0066] SAM은 도 4에 나타낸 것과 같이 USIM 형태로 제공되는 하드웨어 모듈이며, SAM에는 해당 카드사의 고유키 값이 포함되어 있다. 현재 버스 단말기에는 지원하는 카드의 종류의 수만큼 SAM 모듈이 탑재되어 있으며, 예컨대 티머니카드 SAM, 이비카드 SAM, 마이비카드 SAM, 및 버스조합 SAM과 같은 것이 있다. SAM에는 고유키 값이 포함되어 있으며, 이 값은 해당 카드의 인증 작업과 같은 보안 기능을 수행할 때 사용된다.

[0067] 본 발명의 일 실시예에서는, C, C++로 구현된 RF 지불 단말기의 카드처리 기능을 스크립트 파일 기반으로 대체하기 위한 방법을 제공한다. 이때 스크립트 파일은 기존 C로 구현된 기능을 동일하게 수행하며, 수행 기능의 핵심 주체가 된다.

[0068] 또한, 본 발명의 일 실시예에서는, C에서 라이브러리를 호출 하듯이 C코드에서 필요 기능의 스크립트 파일을 호출할 수 있다. 호출된 스크립트 파일은 기존의 C코드와 동일한 기능을 수행할 수 있다. <표 4> 및 <표 5>에 나타낸 소스코드는 C에서 스크립트 파일을 호출하는 함수의 예를 나타낸 것이다(<표 5>는 <표 4>로부터 이어지는 코드이다). <표 4> 및 <표 5>에 나타낸 execJSScript()라는 함수의 인자에, 수행하고자 하는 자바스크립트의 파일명을 적어주면 해당 스크립트가 수행될 수 있다. 이를 통해 원하는 스크립트 파일을 C에서 실행할 수 있다.

표 4

[0069]

```
366 int execJSScript( char *file_name )
367 {
368     FILE *fp;
369     char *jsBuf;
370 
371     JSRuntime *rt;
372     JSContext *ctx;
373     JSObject *global;
374     JSBool ok;
375     jsval rval;
376 
377     // Create a Runtime
378     rt = JS_NewRuntime(1024 * 1024);
379     if(rt == NULL) {
380         printf("rt error...\n");
381     }
382 
383     // Create a Context
384     ctx = JS_NewContext(rt, 4096);
385     if(ctx == NULL) {
386         printf("ctx error...\n");
387     }
388 
389     // Create a Global Object
390     global = JS_NewObject(ctx, NULL, NULL, NULL);
391     if(global == NULL) {
392         printf("global error...\n");
393     }
394 
```

표 5

[0070]

```
395     // Init
396     JS_InitStandardClasses(ctx, global);
397
398     // load a script file.
399     fp = fopen(file_name, "r");
400
401     // get a size of script file.
402     fseek( fp, 0, SEEK_END );
403     uintN size = ftell(fp);
404     fseek( fp, 0, SEEK_SET );
405
406     jsBuf = malloc(size);
407     memset(jsBuf, '₩0', size);
408
409     // read a script file
410     fread( jsBuf, size, 1, fp);
411
412     JS_DefineFunction(ctx, global, "printTime", PrintTime, 1, 0);
413     JS_DefineFunction(ctx, global, "print", Print, 1, 0);
414
415     // Execution
416     ok = JS_EvaluateScript(ctx, global, jsBuf, size, NULL, 0, &rval);
417
418     JS_DestroyContext(ctx);
419     JS_DestroyRuntime(rt);
420     JS_ShutDown();
421
422     free(jsBuf);
423     fclose(fp);
424
425     return 0;
426 }
```

[0071] 본 발명의 실시예에 따르면, 단말기 어플리케이션의 개발 시 기능의 변경 및 추가 등의 작업을 유연하게 처리할 수 있다. 즉, 기존 RF 결재 단말기처럼 정해진 특정 기능만을 수행하는 것이 아니라, 예컨대, 지불 단말기에서도 충전 스크립트를 내려 받으면 충전 단말기로도 사용될 수 있고 조회 단말기로도 사용될 수 있다. 그 결과 단말기의 기능이 유연하게 정의될 수 있으며, 따라서 다양하게 변하는 비즈니스 환경에 맞게 RF 결재 단말기들을 사용할 수 있다.

[0072] <실시예-단말기 제어방법>

[0073] 이하, 본 발명의 일 실시예에 따른 단말기 제어방법을 설명한다.

[0074] 이 제어방법은 카드처리 오브젝트를 제공하는 스크립트 엔진(20)을 포함하는 단말기(1)에서 수행될 수 있다. 이때 카드처리 오브젝트는 예컨대 상술한 RF 오브젝트(211) 또는 SAM 오브젝트(221)와 같은 것일 수 있다. 이 방법은 카드처리 오브젝트를 호출하는 코드를 포함하는 스크립트(711, 712, 또는 713)를 로드하는 단계(S32)를 포함할 수 있다. 위의 스크립트는 예컨대 상술한 <표 1>에 따른 스크립트와 같은 방식으로 제공될 수 있다. 또한 상기 카드처리 오브젝트를 호출하는 코드는, 예컨대 <표 1>에 나타낸 RfDevInit( ), lapdu( )와 같은 것일 수 있다. 그 다음, 이 방법은 스크립트 엔진(20)을 이용하여 상기 스크립트(711, 712, 또는 713)를 실행함으로써 단말기(1)의 디바이스 드라이버를 제어하는 단계를 포함할 수 있다. 이때, 상기 디바이스 드라이버는, 예컨대

상술한 RF 디바이스 드라이버(212) 또는 SAM 디바이스 드라이버(222)와 같은 것일 수 있다.

[0075] 이때, 상기 카드처리 오브젝트는 상기 디바이스 드라이버를 제어하기 위한 함수를 포함할 수 있다. 예컨대, 이 함수는 <표 2>에 나타낸 TRfDevInit( ), <표 3>에 나타낸 RfSCSendApdu( )와 같은 함수일 수 있다. 이때, 이러한 함수는 컴파일 언어로 기술되어 있는 것일 수 있다.

[0076] 또한, 위의 방법은, 스크립트(711, 712, 또는 713)를 로드하는 단계 이전에, 단말기(1)를 초기화하는 단계, 및 서버(2)로부터 스크립트(711, 712, 또는 713)를 제공받는 단계를 더 포함할 수 있다. 즉, 상술한 바와 같이 스크립트(711, 712, 또는 713)를 외부로부터 제공받을 수 있다.

[0077] 이때, 스크립트(711, 712, 또는 713)는, 단말기(1)에 이미 저장되어 있거나, 서버(2)에 의해 제공되는 복수 개의 스크립트 중 단말기(1)가 수행할 어플리케이션에 따라 선택된 것일 수 있다. 예컨대, 단말기(1)가 카드 발급 어플리케이션을 수행하도록 되어 있는 단말기라고 가정하면, 1) 지불 스크립트, 2) 충전 스크립트, 및 3) 잔액 확인 스크립트와 같은 복수 개의 스크립트 중 2) 충전 스크립트가 선택되어 메모리 또는 스크립트 엔진(20)에 로드될 수 있다. 이때, 이 단말기 제어방법은 상기 스크립트를 실행한 결과를 어플리케이션에 리턴 하는 단계를 더 포함할 수 있다.

[0078] <실시예-단말기>

[0079] 본 발명의 일 실시예에 따른 단말기는 상술한 단말기 제어방법에 대한 실시예를 수행하도록 되어 있는 단말기에 관한 것이다.

[0080] 이 단말기는, 카드처리 오브젝트를 제공하는 스크립트 엔진(20), 단말기(1)의 디바이스를 위한 디바이스 드라이버(예: RF 디바이스 드라이버, SAM 디바이스 드라이버), 메모리, 및 처리부를 포함할 수 있다. 이때, 처리부는, 상기 카드처리 오브젝트를 호출하는 코드를 포함하는 스크립트를 상기 메모리에 로드하도록 되어 있고, 상기 스크립트 엔진을 이용하여 상기 스크립트를 실행함으로서 상기 디바이스 드라이버를 제어하도록 되어 있다. 여기서 처리부는 단말기(1)에 포함된 연산장치를 의미할 수 있다. 이 밖에, 이 단말기(1)는 상술한 단말기 제어방법에 대한 실시예에 기재된 각종 기능을 수행할 수 있다.

[0081] 본 발명은 상술한 실시예에 의해 한정되지 않는다.

[0082] 이상 본 발명의 바람직한 실시예와 관련하여 설명하였으나, 본 발명의 기술 분야에 속하는 통상적인 지식을 가진 자들은 본 발명의 본질적인 특성에서 벗어나지 않는 범위 내에 다양한 변경 및 수정을 통하여 용이하게 실시할 수 있을 것이다. 특허청구범위의 각 청구항의 내용은 본 명세서를 통해 이해할 수 있는 범위 내에서 인용관계가 없는 다른 청구항에 결합될 수 있다.

[0083] 그러므로 개시된 실시예는 한정적인 관점이 아니라 설명적인 관점에서 고려되어야 하고, 본 발명의 진정한 범위는 전술한 설명이 아니라 특허청구범위에 나타나 있으며, 그와 동등한 범위 내에 있는 모든 차이점은 본 발명에 포함된 것으로 해석되어야 할 것이다.

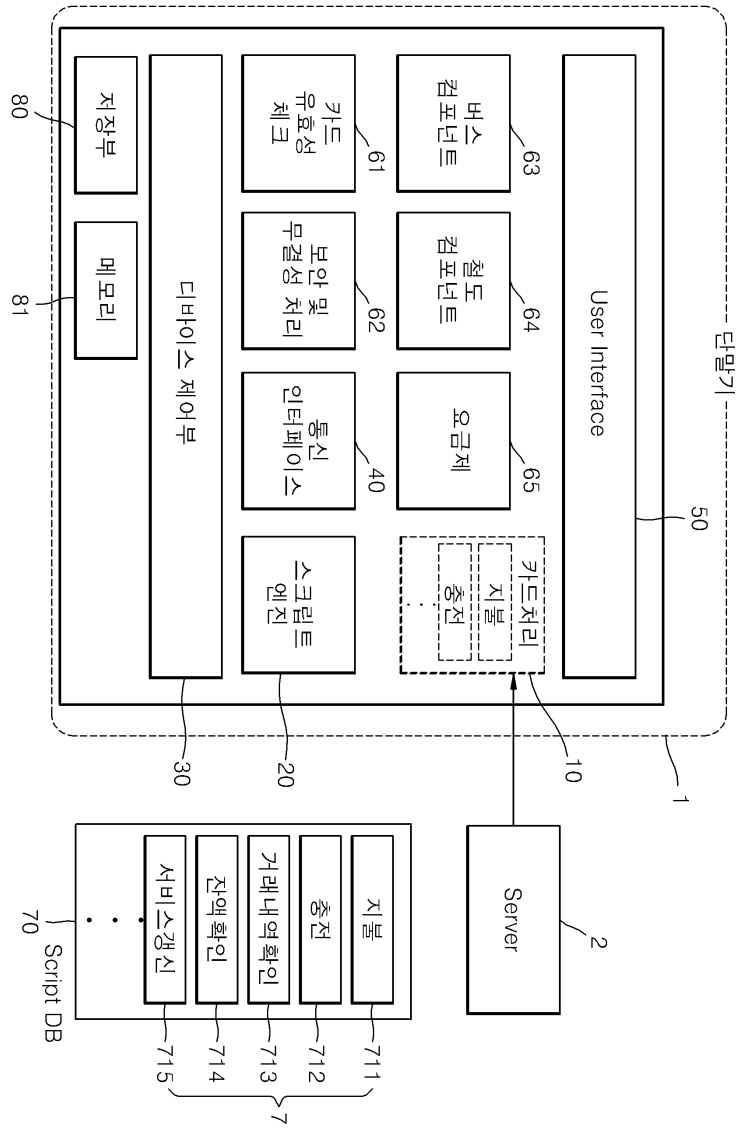
부호의 설명

[0084]

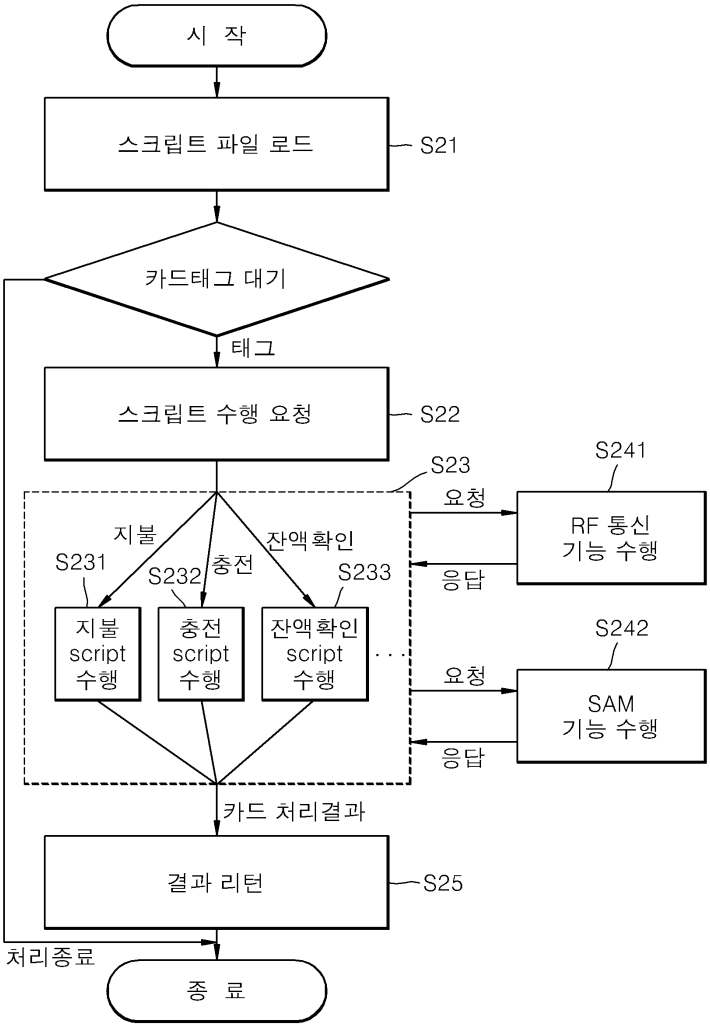
1: 단말기	2: 서버
3: 카드	10: 카드 처리부
20: 스크립트 엔진	71: 스크립트 파일
80: 저장부	100: APDU
211: RF 오브젝트	221: SAM 오브젝트

도면

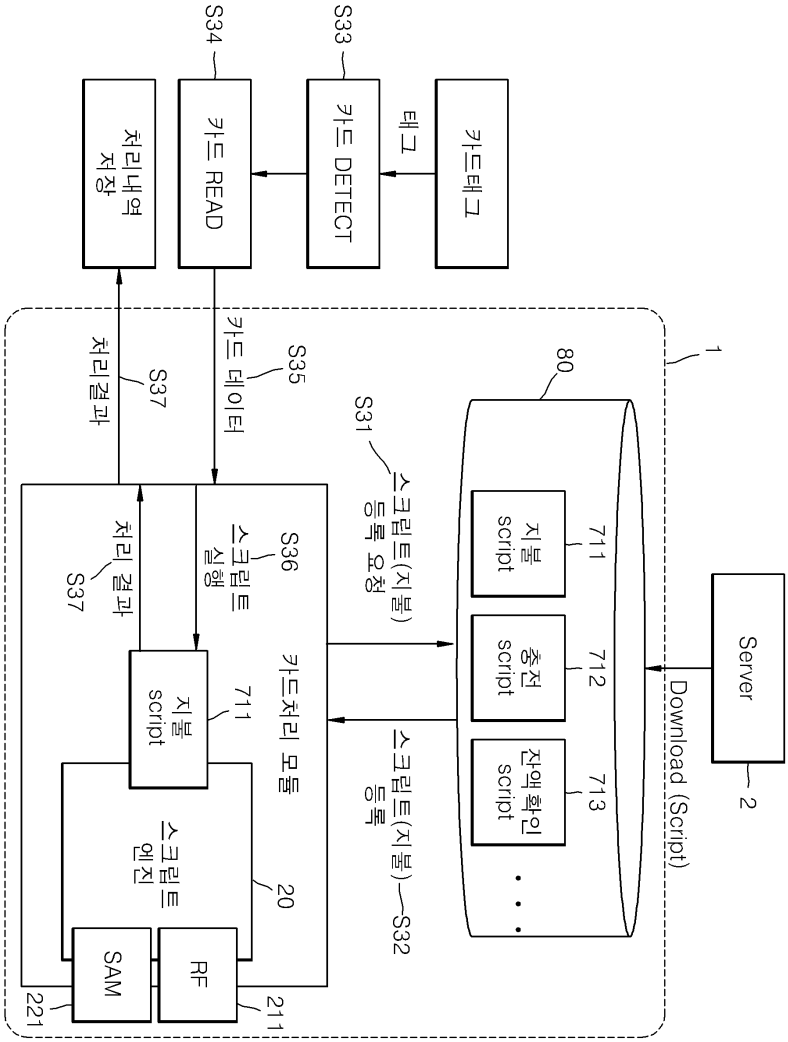
도면1


단말기
1
User Interface
50
버스 컴포넌트
63
철도 컴포넌트
64
요금제
65
카드처리
지불
충전
카드 유효성 체크
61
보안 및 무결성 처리
62
통신 인터페이스
40
스크립트 엔진
20
디바이스 제어부
30
저장부
80
메모리
81
10
Server
2
Script DB
70
서비스갱신
잔액 확인
거래내역확인
충전
지불
715
714
713
712
711
7

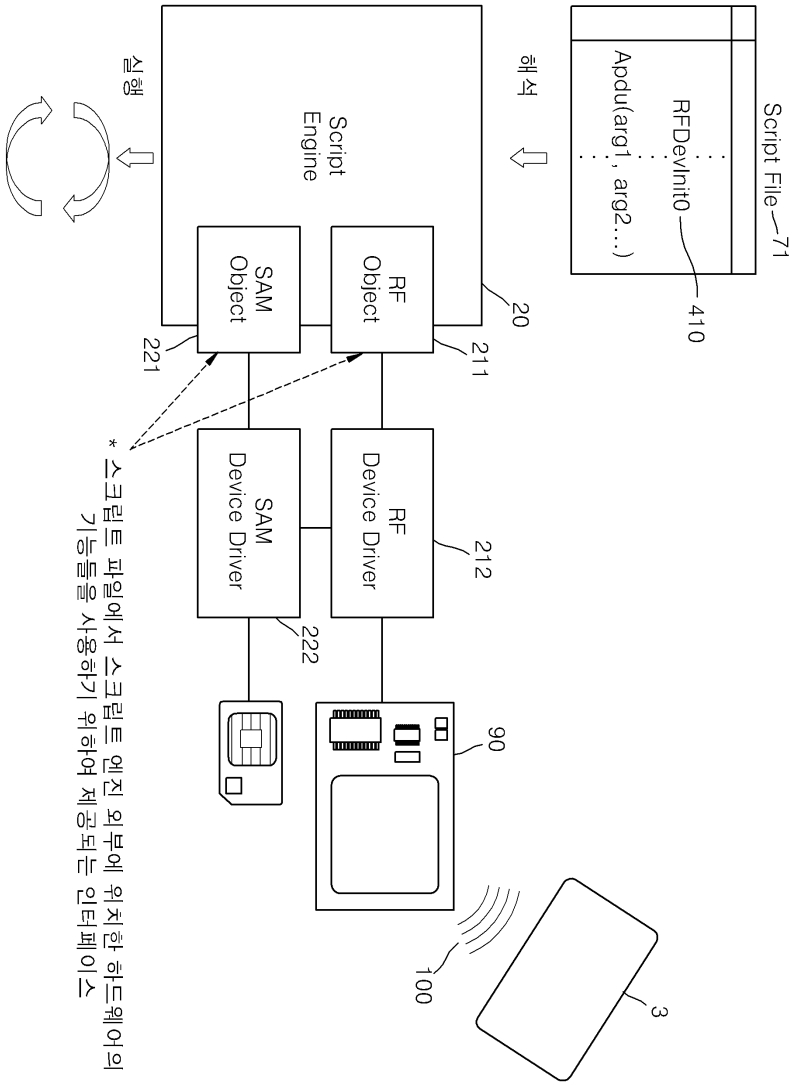
도면2


시 작
스크립트 파일 로드
S21
카드태그 대기
태그
스크립트 수행 요청
S22
S23
S241
요청
RF 통신
기능 수행
응답
지불
잔액확인
충전
S231
S232
S233
지불
script
수행
충전
script
수행
잔액확인
script
수행
S242
요청
SAM
기능 수행
응답
카드 처리결과
결과 리턴
S25
처리종료
종 료

도면3


S34
S33
처리내역 저장
카드 READ
카드 DETECT
카드태그
태그
처리결과
S37
카드 데이터
S35
S31
스크립트(지불) 등록 요청
80
1
711
지불 script
712
충전 script
713
잔액확인 script
Server
2
Download (Script)
처리 결과
S37
S36
스크립트 실행
카드처리 모듈
711
지불 script
스크립트 등록
S32
스크립트 엔진
20
SAM
RF
221
211

도면4


Script File
71
RFDevInit0
Apdu(arg1, arg2...)
410
해석
실행
Script
Engine
20
RF
Object
211
SAM
Object
221
RF
Device Driver
212
SAM
Device Driver
222
90
100
3
* 스크립트 파일에서 스크립트 엔진 외부에 위치한 하드웨어의
기능들을 사용하기 위하여 제공되는 인터페이스

도면5

명령 APDU
명령 APDU 헤더
CLA
INS
P1
P2
Lc
Data
Le
CLA: 클래스 바이트(commend-ID), INS: 명령어 코드, P1, P2: 파라미터, Lc: 명령 APDU 데이터 길이.
Data: 명령 APDU 데이터, Le: 응답 APDU상의 데이터 길이
응답 APDU
Data
SW1
SW2
Data: 응답 APDU 데이터, SW1, SW2: 카드 상태 코드