



(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。 Int. Cl.

H04L 29/02 (2006.01)

H04L 29/06 (2006.01)

H04Q 7/20 (2006.01)

(45) 공고일자

2007년02월15일

(11) 등록번호

10-0683061

(24) 등록일자

2007년02월08일

(21) 출원번호

10-2005-0101776

(65) 공개번호

(22) 출원일자

2005년10월27일

(43) 공개일자

심사청구일자

2005년10월27일

(73) 특허권자

주식회사 케이티프리텔
서울 송파구 신천동 7-18

(72) 발명자

안병익
경기도 고양시 일산구 마두동 정발마을6단지 602동 302호

차경찬

인천광역시 연수구 동춘동 대우삼환아파트 105동 505호

(74) 대리인

유미특허법인

(56) 선행기술조사문헌

KR1020000056396 A

KR1020050059185 A

KR1020060056411 A

KR1020060066021 A

* 심사관에 의하여 인용된 문헌

심사관 : 인치복

전체 청구항 수 : 총 10 항

(54) 모바일 플랫폼의 수정없이 신규 모바일 기능을 추가하는 방법, 이를 적용한 모바일 장치 및 컴퓨터로 판독가능한 기록매체

(57) 요약

본 발명은 모바일 플랫폼의 수정없이 신규 모바일 기능을 추가하는 방법, 이를 적용한 모바일 장치 및 컴퓨터로 판독가능한 기록 매체에 관한 것이다.

본 발명에 의하면 어플리케이션 모듈이 기능 모듈을 이용하기 위한 모바일 기능에 대한 함수의 정의 및 프로토 타입을 제공하는 API 모듈을 포함한 모바일 플랫폼이 구비된 모바일 장치에 대한 모바일 기능의 추가 방법에 있어서, a) 선택재된 모바일 플랫폼에 새로운 모바일 기능을 정의한 함수 및 프로토 타입을 제공하는 API 모듈을 상기 모바일 플랫폼에 추가하는 단계; b) 상기 API 모듈과 대응하는 기능 모듈을 추가하는 단계; c) 선택재된 모바일 플랫폼에 새로운 모바일 기능의 추가 및 구현을 상기 모바일 플랫폼과 독립적으로 수행하는 함수로 구성된 터널링 모듈에 의하여 상기 API 모듈과 대응하는 상기 기능 모듈을 호출하는 단계; 및 d) 상기 터널링 모듈을 이용하여 상기 기능 모듈이 제공하는 기능을 상기 API 모듈로 전달하는 단계를 포함한다.

대표도

도 2

특허청구의 범위

청구항 1.

모바일 기능에 대한 함수의 정의 및 프로토 타입을 제공하는 API 모듈을 포함한 모바일 플랫폼이 구비된 모바일 장치에 모바일 기능을 추가하는 방법에 있어서,

- a) 선택재된 모바일 플랫폼에 새로운 모바일 기능을 정의한 함수 및 프로토 타입을 제공하는 API 모듈을 상기 모바일 플랫폼에 추가하는 단계;
 - b) 상기 모바일 플랫폼에 추가한 API 모듈에 정의된 모바일 기능을 구현하는 기능 모듈을 모바일 장치에 추가하는 단계;
 - c) 선택재된 모바일 플랫폼에 새로운 모바일 기능의 추가 및 구현을 상기 모바일 플랫폼과 독립적으로 수행하는 함수로 구성되어 상기 모바일 플랫폼에 탑재된 터널링 모듈을 이용하여 상기 API 모듈에 정의된 모바일 기능을 구현하는 기능 모듈을 호출하는 단계; 및
 - d) 상기 터널링 모듈을 이용하여 상기 기능 모듈이 제공하는 기능을 상기 API 모듈로 전달하는 단계
- 를 포함하는 모바일 기능의 추가 방법.

청구항 2.

제1항에 있어서, 상기 a)단계 이전에

- i) 상기 터널링 모듈을 상기 모바일 플랫폼에 정의하는 단계;
 - ii) 상기 터널링 모듈을 적용할 수 있는 모바일 기능을 상기 터널링 모듈에 정의하는 단계
- 를 더 포함하는 모바일 기능의 추가 방법.

청구항 3.

제2항에 있어서, 상기 ii)단계 이후

- iii) 추가할 모바일 기능이 상기 터널링 모듈에 정의된 모바일 기능인지 여부를 판단하는 단계; 및
 - iv) 상기 판단 결과 상기 터널링 모듈에 정의된 모바일 기능에 대해서 추가를 결정하는 단계
- 를 더 포함하는 모바일 기능의 추가 방법.

청구항 4.

제2항 또는 제3항에 있어서, 상기 모바일 기능에 대해 상기 터널링 모듈의 적용 대상 여부는 상기 터널링 모듈에 정의된 상기 모바일 기능을 식별하는 식별자를 이용하는 것임을 특징으로 하는 모바일 기능의 추가 방법.

청구항 5.

모바일 응용 프로그램을 구동하는 어플리케이션 모듈에게 모바일 기능을 구현하는 기능 모듈이 제공하는 모바일 기능을 이용할 수 있는 환경을 제공하는 모바일 플랫폼이 구비된 모바일 장치에 있어서,

상기 기능 모듈이 제공하는 모바일 기능을 어플리케이션 모듈이 이용하기 위한 함수들을 정의하고 프로토 타입을 제공하는 API를 보관하는 제1 저장부;

상기 제1 저장부에 보관된 API에 새로운 모바일 기능을 이용하기 위한 함수들을 정의하고 프로토 타입을 제공하는 추가된 API를 보관하는 제2 저장부;

모바일 플랫폼에 새로운 모바일 기능의 추가 및 구현을 상기 모바일 플랫폼과 독립적으로 수행하는 함수로 구성된 터널링 API를 보관하는 제3 저장부;

상기 제1 저장부 및 제2 저장부에 보관된 API의 다운로드/실행/저장/삭제 등을 관리하고 상기 API에 정의된 모바일 기능을 구현하는 각각의 기능 모듈의 호출 및 상기 기능 모듈로부터 기능을 전달받는 동작을 수행하며, 상기 제2 저장부에 보관된 API와 이와 대응하는 기능 모듈에 대해서는 상기 제3 저장부에 저장된 터널링 API에 의한 터널링 기능을 적용하는 제어 모듈; 및

상기 제1 저장부 및 제2 저장부에 보관된 API에 의해 정의된 기능을 구현하는 기능 모듈

를 포함하는 모바일 장치.

청구항 6.

제5항에 있어서, 상기 제어 모듈은

상기 제1 저장부 및 제2 저장부에 API의 다운로드/실행/저장/삭제등을 관리하고 상기 제1 저장부에 보관된 API가 정의하는 기능을 구현하는 기능 모듈을 호출하여 이로부터 기능을 전달받는 동작을 수행하는 API 관리자 모듈; 및

상기 제3 저장부에 보관된 터널링 API가 정의하는 터널링 기능을 수행하고 상기 터널링 기능을 통하여 상기 제2 저장부에 보관된 API가 정의하는 기능을 구현하는 기능 모듈을 호출하여 이로부터 기능을 전달받는 동작을 수행하는 터널링 관리자 모듈

을 포함하는 모바일 장치.

청구항 7.

제6항에 있어서, 상기 터널링 관리자 모듈은

상기 터널링 기능을 적용 가능한 모바일 기능을 정의하는 정의부;

상기 정의부에 정의된 사항에 따라 상기 API 관리자 모듈에 의한 상기 제2 저장부에 API의 저장 여부 및 이와 대응하는 기능 모듈의 추가 여부를 판단하는 판단부;

상기 판단부의 판단 결과에 따라 상기 제2 저장부에 저장된 API와 대응하는 기능 모듈을 호출하는 호출부; 및

상기 호출부에 의해 호출된 기능 모듈로부터 상기 기능 모듈이 제공하는 기능을 전달받는 전달부

를 포함하는 모바일 장치.

청구항 8.

제5항에 있어서, 상기 제3 저장부에 보관된 터널링 API는

상기 어플리케이션 모듈이 상기 제2 저장부에 보관된 API와 대응하는 기능 모듈과 통신 하기 위한 구조, 상기 기능 모듈이 제공하는 기능이 담긴 데이터를 전달받기 위한 이벤트 및 상기 이벤트를 상기 기능 모듈로부터 전달받는 구조를 정의한 함수로 구성되는 것을 특징으로 하는 모바일 장치.

청구항 9.

제8항에 있어서, 상기 제3 저장부에 보관된 터널링 API는

상기 터널링 API를 이용할 수 있는 기능 모듈의 종류를 정의하는 구조 및 상기 어플리케이션 모듈이 호출하고자 하는 기능 모듈이 정의된 것인지 여부를 판단하는 구조를 정의한 함수를 더 포함하는 것을 특징으로 하는 모바일 장치.

청구항 10.

어플리케이션 모듈이 기능 모듈을 이용하기 위한 모바일 기능에 대한 함수의 정의 및 프로토 타입을 제공하는 API 모듈을 포함한 모바일 플랫폼이 구비된 모바일 장치에 대해 모바일 기능의 추가 방법에 관한 프로그램이 저장된 컴퓨터로 판독가능한 기록매체에 있어서,

- a) 선택재된 모바일 플랫폼에 새로운 모바일 기능을 정의한 함수 및 프로토 타입을 제공하는 API 모듈을 상기 모바일 플랫폼에 추가하는 기능;
- b) 상기 API 모듈과 대응하는 기능 모듈을 추가하는 기능;
- c) 선택재된 모바일 플랫폼에 새로운 모바일 기능의 추가 및 구현을 상기 모바일 플랫폼과 독립적으로 수행하는 함수로 구성된 터널링 모듈에 의하여 상기 API 모듈과 대응하는 상기 기능 모듈을 호출하는 기능; 및
- d) 상기 터널링 모듈을 이용하여 상기 기능 모듈이 제공하는 기능을 상기 API 모듈로 전달하는 기능

를 포함하는 모바일 기능의 추가 방법에 관한 프로그램이 저장된 컴퓨터로 판독가능한 기록매체.

명세서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 모바일 플랫폼의 수정없이 신규 모바일 기능을 추가하는 방법, 이를 적용한 모바일 장치 및 기록 매체에 관한 것이다.

일반적으로 모바일은 휴대폰과 휴대용 개인 정보 단말기(PDA) 등과 같이 이동성을 가진 수단을 총칭하는 것으로 손으로 들고 다니므로 가볍고 작은 것이 특징이다. 하드웨어, 소프트웨어 기술의 발전에 따라 모바일 단말의 기능이 기존의 음성 호 처리 및 단문 전송 데이터 처리 외에 게임, 일정 관리, 계산기, 알람 등 이용자에게 유용한 부가 기능이 탑재되고 있으며

또한 이동통신단말에서 인터넷 접속이 가능해지면서 인터넷을 통하여 필요한 정보를 검색하고 콘텐츠 및 서비스를 이용할 수 있게 되었다. 현재 이동통신 단말기에 포함되어 있는 부가 기능은, 국제시각 확인기능, 통화시간 확인기능, 전화번호 메모리기능, 애니메이션 대기화면, 최근 발신/수신번호 보기, 지역번호/국제전화코드 검색, 세계시계 기능, 알람 설정기능, 개인정보 관리기능, 단축다이얼/음성다이얼 관리, 스케줄(일간/주간/월간) 관리, 이메일(E-mail) 송수신 기능, 웹 브라우저(Wap-Browser) 기능, 영한/한영 사전 기능, 계산기 기능, 게임 기능 등이 있다. 불과 5년 전만 해도 휴대폰은 단지 이동 중에 전화를 하거나 간단한 문자 몇 마디를 주고 받을 수 있는 통신 수단에 불과했다. 그러나 지금은 우리 사회의 문화 아이콘으로 자리 잡을 정도로 그 기능이 다양해졌다.

따라서, 초기 탑재된 모바일 기능 외에 추가적으로 생성되는 모바일 기능의 종류가 다양해짐에 따라 새로운 모바일 기능의 생성시 즉각 모바일 장치에 반영할 필요가 있다.

그런데, 종래에는 기존의 모바일 기능 외에 새로운 모바일 기능을 탑재하고자 할 경우 새로운 모바일 기능을 구현할 수 있도록 기존의 모바일 플랫폼을 수정해야 한다.

즉, 첨부된 도 1에 보인 바와 같이, 기존의 모바일 플랫폼에 새로운 모바일 기능을 탑재하고자 할 경우 이를 구현하는 API(Application Programming Interface, 이하 약어로 기술)모듈 및 기능 모듈을 추가하고 이에 적합하도록 모바일 플랫폼을 새로이 정의해야 했다.

발명이 이루고자 하는 기술적 과제

따라서, 본 발명은 상기와 같은 문제점을 감안하여 안출된 것으로 모바일 플랫폼의 수정없이 기존에 탑재된 모바일 기능 외에 모바일 기능의 추가 탑재가 용이하도록 하는 터널링 기능을 이용하여 신규 모바일 기능을 추가하는 방법, 이를 적용한 모바일 장치 및 기록 매체를 제공한다.

발명의 구성

상기 기술한 바와 같은 과제를 이루기 위하여 본 발명의 하나의 특징에 의하면 모바일 기능에 대한 함수의 정의 및 프로토타입을 제공하는 API 모듈을 포함한 모바일 플랫폼이 구비된 모바일 장치에 모바일 기능을 추가하는 방법에 있어서, a) 선택된 모바일 플랫폼에 새로운 모바일 기능을 정의한 함수 및 프로토타입을 제공하는 API 모듈을 상기 모바일 플랫폼에 추가하는 단계; b) 상기 모바일 플랫폼에 추가한 API 모듈에 정의된 모바일 기능을 구현하는 기능 모듈을 모바일 장치에 추가하는 단계; c) 선택된 모바일 플랫폼에 새로운 모바일 기능의 추가 및 구현을 상기 모바일 플랫폼과 독립적으로 수행하는 함수로 구성되어 상기 모바일 플랫폼에 탑재된 터널링 모듈을 이용하여 상기 API 모듈에 정의된 모바일 기능을 구현하는 기능 모듈을 호출하는 단계; 및 d) 상기 터널링 모듈을 이용하여 상기 기능 모듈이 제공하는 기능을 상기 API 모듈로 전달하는 단계를 포함한다.

본 발명의 또 다른 특징에 의하면 모바일 응용 프로그램을 구동하는 어플리케이션 모듈에게 모바일 기능을 구현하는 기능 모듈이 제공하는 모바일 기능을 이용할 수 있는 환경을 제공하는 모바일 플랫폼이 구비된 모바일 장치에 있어서, 상기 기능 모듈이 제공하는 모바일 기능을 어플리케이션 모듈이 이용하기 위한 함수들을 정의하고 프로토타입을 제공하는 API를 보관하는 제1 저장부; 상기 제1 저장부에 보관된 API에 새로운 모바일 기능을 이용하기 위한 함수들을 정의하고 프로토타입을 제공하는 추가된 API를 보관하는 제2 저장부; 모바일 플랫폼에 새로운 모바일 기능의 추가 및 구현을 상기 모바일 플랫폼과 독립적으로 수행하는 함수로 구성된 터널링 API를 보관하는 제3 저장부; 상기 제1 저장부 및 제2 저장부에 보관된 API의 다운로드/실행/저장/삭제 등을 관리하고 상기 API에 정의된 모바일 기능을 구현하는 각각의 기능 모듈의 호출 및 상기 기능 모듈로부터 기능을 전달받는 동작을 수행하며, 상기 제2 저장부에 보관된 API와 이와 대응하는 기능 모듈에 대해서는 상기 제3 저장부에 저장된 터널링 API에 의한 터널링 기능을 적용하는 제어 모듈; 및 상기 제1 저장부 및 제2 저장부에 보관된 API에 의해 정의된 기능을 구현하는 기능 모듈을 포함한다.

아래에서는 첨부한 도면을 참고로 하여 본 발명의 실시예에 대하여 본 발명이 속하는 기술 분야에서 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. 그리고 도면에서 본 발명을 명확하게 설명하기 위해서 설명과 관계없는 부분은 생략하였으며, 명세서 전체를 통하여 유사한 부분에 대해서는 유사한 도면 부호를 붙였다. 또한, 어떤 부분이 어떤 구성요소를 "포함"한다고 할 때, 이는 특별히 반대되는 기재가 없는 한 다른 구성요소를 제외하는 것이 아니라 다른 구성요소를 더 포함할 수 있는 것을 의미한다. 또한, 본 명세서에서 기재한 모듈(module)이란 용어는 특정한 기능이나 동작을 처리하는 하나의 단위를 의미하며, 이는 하드웨어나 소프트웨어 또는 하드웨어 및 소프트웨어의 결합으로 구현할 수 있다.

이제 본 발명의 실시예에 따른 모바일 플랫폼의 수정없이 신규 모바일 기능을 추가하는 방법, 이를 적용한 모바일 장치 및 컴퓨터로 판독가능한 기록매체에 대하여 도면을 참고로 하여 상세하게 설명한다.

도 2는 본 발명에 따른 신규 모바일 기능을 추가하는 개념도이다. 상기 도면에 보인 바와 같이, 모바일 플랫폼에 탑재된 터널링 모듈을 이용하여 상기 모바일 플랫폼에 정의되어 있지 않은 새로운 모바일 기능을 상기 모바일 플랫폼에 영향을 주지 않고 추가할 수 있다. 여기서, 터널링 모듈은 모바일 기능의 추가시마다 이를 지원할 수 있도록 모바일 플랫폼의 전체 소스 코드를 수정할 필요없이 추가되는 모바일 기능과 관련한 부분을 담당하는 터널링 기능을 정의한 함수로 구성된다. 따라서, 도 1에 보인 바와 같이 신규 모바일 기능을 추가할 때마다 이에 적합하도록 모바일 플랫폼을 수정해야 했던 종래와 달리 상기 모바일 플랫폼의 수정이 필요없는 것이다.

도 3은 본 발명에 따른 터널링 기능을 보인 개념도이다. 즉, 상기 도 2에 보인 터널링 기능을 구체화한 도면으로 상기 터널링 기능을 이용하여 기존의 모바일 플랫폼이 제공하는 모바일 기능 외에 추가로 모바일 기능을 탑재하는 경우를 보인 것이다. 상기 도면에 보인 바와 같이, 추가되는 모바일 기능을 정의한 함수 및 프로토 타입을 제공하는 신규 API 모듈을 추가하고 이와 대응하는 즉 추가되는 모바일 기능을 구현하는 기능 모듈을 추가한다. 상기 추가된 신규 API 모듈은 터널링 기능을 정의한 함수 및 프로토 타입을 제공하는 터널링 API를 이용하여 해당하는 기능 모듈을 호출하는 파라미터를 전달한다. 상기 파라미터를 전달받은 기능 모듈은 결과값(상기 파라미터에 대한 결과값으로서 상기 기능 모듈이 제공하는 기능이 담긴 데이터)을 상기 터널링 API를 이용하여 전달한다. 이와 같이, 상기 터널링 기능은 추가된 API 모듈과 기능 모듈 간에 상호 연계를 지원한다.

이하, 상기 터널링 기능의 개념이 적용된 신규 모바일 기능의 추가 방법 및 모바일 장치를 설명하기로 한다.

도 4는 본 발명의 실시예에 따른 신규 모바일 기능을 추가하는 방법을 보인 순서도이다. 상기 도면에 보인 바와 같이, 신규 모바일 기능을 추가하는 방법은 API 모듈 추가 단계(S110), 기능 모듈 추가 단계(S130), 터널링 모듈을 이용한 기능 모듈 호출 단계(S150) 및 터널링 모듈을 이용한 기능 전달 단계(S170)를 포함한다.

상기 API 모듈 추가 단계(S110)는 기존의 모바일 플랫폼이 제공하는 모바일 기능 이외의 추가되는 새로운 모바일 기능을 구현하기 위한 함수 및 프로토 타입을 제공하는 API 모듈을 상기 모바일 플랫폼에 추가하는 단계이다.

상기 기능 모듈 추가 단계(S130)는 상기 단계(S110)에서 추가된 API 모듈과 대응되는 기능 모듈로서 상기 추가되는 새로운 모바일 기능을 구현하는 기능 모듈을 추가하는 단계이다.

상기 터널링 모듈을 이용한 기능 모듈 호출 단계(S150)는 상기 도 2 및 도 3에서 설명한 터널링 기능을 구현하기 위한 함수를 제공하는 터널링 모듈을 이용하여 상기 API 모듈이 해당하는 기능 모듈을 호출하는 단계이다. 보다 상세하게는 어플리케이션 모듈이 상기 API 모듈에 정의된 API를 이용하여 상기 기능 모듈이 제공하는 기능을 이용하기 위해 상기 터널링 모듈을 통하여 상기 기능 모듈을 호출하는 단계이다. 여기서, 상기 어플리케이션 모듈은 API 모듈에 정의된 함수를 응용하여 작성된 모바일 응용 프로그램을 지칭한다.

상기 터널링 모듈을 이용한 기능 전달 단계(S170)는 상기 단계(S170)에서 호출한 기능 모듈로부터 상기 기능 모듈이 제공하는 기능을 상기 터널링 모듈을 통하여 전달받는 단계이다. 보다 상세하게는 상기 어플리케이션 모듈이 상기 터널링 모듈을 통하여 상기 기능 모듈이 제공하는 기능을 이용하는 단계이다.

도 5는 본 발명의 또 다른 실시예에 따른 신규 모바일 기능을 추가하는 방법을 보인 순서도로서, 상기 도 4에서 API 모듈을 추가하기에 앞서 몇 가지 단계를 더 포함한다. 즉 터널링 모듈의 정의 단계(S101) 및 터널링 모듈의 적용 대상 판단 단계(S103)를 더 포함한다.

상기 터널링 모듈의 정의 단계(S101)는 상기 도 2 및 도 3에서 설명한 터널링 기능을 구현하기 위한 터널링 모듈을 모바일 플랫폼에 정의하는 단계이다.

이때, 상기 터널링 기능의 적용 대상이 되는 기능 모듈을 정의하며 상기 정의 절차는 상기 기능 모듈의 식별자를 이용하는 것이 바람직하다.

상기 터널링 모듈의 적용 대상 판단 단계(S103)는 추가하고자 하는 모바일 기능을 구현하는 기능 모듈이 상기 단계(S101)에서 정의된 것인지 여부를 판단하고 상기 판단 결과 상기 단계(S101)에서 정의된 기능 모듈과 이에 대응하는 API 모듈만을 추가하도록 결정하는 단계이다.

도 6은 본 발명의 또 다른 실시예에 따른 신규 모바일 기능을 추가하는 일련의 과정을 보인 흐름도로서, 상기 도 4 및 도 5를 구체적으로 구현하는 과정을 보인 것이다.

상기 도면에 보인 바와 같이 터널링 기능을 통하여 신규 탑재된 모바일 기능을 이용하는 일련의 흐름은 터널링 기능 적용 여부 판단 단계(S200), 기능 모듈의 인스턴스 생성 단계(S210), 터널링 기능을 이용한 기능 모듈 호출 단계(S220) 및 터널링 기능을 이용한 데이터 전달 단계(S230, S2400)로 이루어진다.

먼저, 상기 터널링 기능 적용 여부 판단 단계(S200)는 추가하고자 하는 기능 모듈이 기존의 모바일 플랫폼을 변경하지 않고 터널링 기능을 통하여 이용할 수 있는 기능 모듈인지 여부를 판단하는 단계이다. 여기서, 기능 모듈은 OEM(Original Equipment Manufacturer)모듈을 지칭하는 것으로서 모바일 장치 제조업자가 제공하는 기능을 구현하는 모듈이다. 이하 도 6을 참조한 본 명세서에서는 상기 기능 모듈을 OEM 모듈이라 기술한다.

상기 신규 탑재된 기능 모듈의 터널링 기능 적용 여부 판단 절차는 어플리케이션 모듈이 모바일 플랫폼으로 상기 신규 탑재된 OEM 모듈의 터널링 기능 적용 여부를 판단할 것을 요청하는 명령어를 전달한다. 이때 사용되는 API 함수는 'MC_gaSupportedOemModule()'이다. 그리고, 상기 명령어를 전달받은 모바일 플랫폼은 해당하는 OEM 모듈로 API 함수 'GetSysInformation():GAMODULE'를 이용하여 상기 OEM 모듈의 ID값(또는 식별자)을 요청한다. 그리고, 상기 OEM 모듈로부터 'ret: GetSysInformation(): ret->OEMModule'을 통하여 상기 ID값(또는 식별자)을 전달받는다. 이때, 상기 OEM 모듈의 ID값(또는 식별자)은 사전에 다음과 같은 OEM 모듈 정의 단계를 거쳐 모바일 플랫폼에 정의한 것이다.

```
Typedef enum _MH_KTFGA_MOD
```

```
{
```

```
KTF_GA_NOTSUPPORT=0,
```

```
KTF_GA_MOD1=1,
```

```
KTF_GA_MOD2=2,
```

```
...
```

```
} MH_KTFGA_MOD
```

그리고, 상기 모바일 플랫폼은 OEM 모듈로부터 전달받은 ID값(또는 식별자)에 해당하는 OEM 모듈이 터널링 기능의 적용 대상인지를 상기 터널링 기능에서 제공하는 API함수를 이용하여 판단하며, 이때 이용하는 API 함수는 다음과 같다.

[사용 함수]

```
M_Boolean MC_gaIsSupportedOemModule(
```

```
M_Int32 moduleID
```

```
);
```

[파라미터]

```
moduleID = KTF_GA_MOD1 // 사용하고자 하는 기능 모듈(즉, OEM 모듈)의 ID 값
```

즉, 상기 신규 탑재된 OEM 모듈의 터널링 기능 적용 여부 판단 단계(S200)는 신규 추가된 OEM 모듈이 상기 OEM 모듈 정의 단계를 통해 정의된 OEM 모듈의 ID값(또는 식별자)과 대응되는지 여부를 판단하는 것이다.

그리고, 상기 신규 탑재된 OEM 모듈의 터널링 기능 적용 여부 판단 결과는 'ret : MC_gaSupportedOemModule(): T or F'를 통하여 모바일 플랫폼으로부터 어플리케이션 모듈로 전달된다.

다음에, 상기 OEM 모듈의 인스턴스 생성 단계(S210)는 상기 신규 탑재된 기능 모듈의 터널링 기능 적용 여부 판단 결과 사용하고자 하는 OEM 모듈이 터널링 기능의 적용 대상인 경우, 해당 OEM 모듈을 오픈하는 단계이다.

상기 어플리케이션 모듈이 'MC_gaInterface(): CMD->GA_OPEN_MOD'를 통하여 모바일 플랫폼에 해당 OEM 모듈의 오픈을 요청한다. 상기 모바일 플랫폼은 보안 레벨의 체크 후 'MH_gaInterface():CMD->GA_OPEN_MOD'를 통하여 상기 OEM 모듈로 오픈을 요청한다. 이때 상기 OEM 모듈의 인스턴스가 지원되는 경우 해당 인스턴스를 리턴하며 사용되는 API 함수는 다음과 같다.

[사용 함수]

M_Int32 MC_gaInterfaceL1~L7(

M_Int32 oemModule,

M_Int32 ModInstanceID,

M_ Char * cmd,

Void * param1,

M_Int32* param1Size

Void* param2 ,

M_Int32* param2Size

);

[파라미터]

[IN] oemModule= GA_KTF_MOD1, // OPEN 하고자 하는 OEM 모듈 ID

[IN] ModInstanceID=-1, // 초기 OPEN 시 -1을 넣는다.

[IN] cmd = "GA_OPEN_MOD", // OEM 모듈 OPEN 을 위한 커맨드

[OUT] param1 , // OPEN 된 OEM 모듈의 INSTANCE ID 값이 리턴된다(1~n)

// INSTANCE를 지원 하지 않을 경우 0을 리턴한다.

[OUT] param1Size // 버퍼의 사이즈

[IN] param2 =NULL, // 디폴트 처리

[IN] param2Size =0; // 디폴트 처리

상기 OEM 모듈의 인스턴스는 상기 OEM 모듈로부터 'MH_gaInterface():param1->modInstance'를 통하여 상기 모바일 플랫폼으로 리턴된다. 그리고 'ret : MC_gaInterface():param1->modInstance'를 통하여 상기 모바일 플랫폼으로부터 상기 어플리케이션 모듈로 리턴된다.

다음에, 상기 터널링 기능을 통한 기능 모듈 호출 단계(S220)는 상기 OEM 모듈을 사용하기 위해 CMD(CoManD)를 통해 해당 HAL(Handset Adaptation Layer, 이하 약어로 기술)을 호출하는 단계이다. 즉, 상기 어플리케이션 모듈이 'MC_gaInterface(): CMD'를 통하여 모바일 플랫폼으로 상기 OEM 모듈과의 통신을 담당하는 해당 HAL을 호출한다. 그리고 상기 모바일 플랫폼은 보안 레벨의 체크 후 상기 OEM 모듈로 'MH_gaInterface(): CMD'를 통하여 상기 HAL을 호출한다. 상기 OEM 모듈은 호출된 HAL정보를 'ret: MH_gaInterface(): param1,param2->Data'를 통하여 모바일 플랫폼으로 전달한다. 일반적으로 상기 HAL은 모바일 장치 또는 기본 모바일 소프트웨어(또는 모바일 운영체제)에 관계없이 API를 구현하기 위한 일종의 장치 드라이버를 지칭하는 것이다. 그런데 여기서 HAL은 특히 상기 OEM 모듈이 제공하는 기능을 이용하기 위한 API를 구현하는 HAL을 지칭한다.

이후, 상기 모바일 플랫폼은 상기 전달받은 HAL 정보를 'ret: MC_gaInterface(): param1,param2->Data'를 통하여 상기 어플리케이션 모듈로 전달한다.

상기 어플리케이션 모듈이 CMD를 통해 해당하는 OEM 모듈을 사용하기 위해 터널링 기능에서 제공하는 API 함수는 다음과 같다.

[사용 함수]

M_Int32 MC_gaInterfaceL1~L7(

M_Int32 oemModule,

M_Int32 ModInstanceID,

M_ Char * cmd,

Void * param1,

M_Int32* param1Size

Void* param2 ,

M_Int32* param2Size

);

[파라미터]

[IN] oemModule= GA_KTF_MOD1, // 사용 하고자 하는 OEM 모듈 ID

[IN] ModInstanceID=1, // 초기 OPEN 시 OEM 모듈에서 생성한 INSTANCE ID값을 넣는다.

[IN] cmd = "MH_A_1", // OEM 모듈 OPEN 을 위한 커맨드

[IN/OUT] param1 , // OPEN 된 OEM 모듈의 입출력을 위한 버퍼

[IN/OUT] param1Size // 버퍼의 사이즈

[IN/OUT] param2 , // OPEN 된 OEM 모듈의 입출력을 위한 버퍼

```
[IN/OUT] param2Size; // 버퍼의 사이즈
```

다음에, 상기 터널링 기능을 통한 데이터 전달 단계(S230, S240)는 어플리케이션 모듈이 터널링 기능을 통하여 OEM 모듈과 통신하여 상기 어플리케이션 모듈이 상기 OEM 모듈이 제공하는 기능을 전달받는 단계이다. 이때 상기 어플리케이션 모듈이 OEM 모듈로부터 데이터를 전달받는 단계는 상기 OEM 모듈의 기능이 blocking mode에서 동작하는 경우(S230)와 non-blocking mode에서 동작하는 경우(S240)로 나누어 질 수 있다.

먼저, 상기 OEM 모듈에서 제공 하는 기능이 blocking mode로 동작 하는 경우(S230) OEM 모듈은 활성화될 수 없으며 (Layer Bottom 유지해야 한다) 상기 OEM 모듈에서는 상기 OEM 모듈이 생성한 데이터만을 플랫폼으로 전달해야 한다.

즉 터널링 기능에서 제공하는 이벤트를 통하여 상기 데이터의 전달이 이루어지는데 먼저 상기 어플리케이션 모듈이 'MC_gaInterface(): CMD->GetData'를 통하여 상기 모바일 플랫폼으로 데이터를 요청한다. 상기 모바일 플랫폼은 보안 레벨의 체크 후 'MH_gaInterface(): CMD->GetData'를 통하여 상기 OEM 모듈로 데이터를 요청한다. 상기 모바일 플랫폼은 상기 OEM 모듈로부터 'ret: MH_gaInterface():ret->True'를 통하여 상기 요청에 대한 응답을 전달받는다. 그리고 상기 전달받은 응답은 'ret: MC_gaInterface():ret->True'를 통하여 상기 어플리케이션 모듈로 전달된다. 그리고 상기 요청한 데이터는 OEM 모듈의 HAL을 통하여 이벤트로 전달한다. 상기 이벤트는 OEM 모듈로부터 모바일 플랫폼으로 전달되는 'MH_pltSendEvent(): MH_KTFUSEROEM_EVENT->SubEvt,Data'와 상기 모바일 플랫폼 내부에 이벤트 큐를 저장하고 상기 어플리케이션 모듈로 전달되는 'MH_KTFUSEROEM_EVENT'를 사용한다. 그리고 상기 어플리케이션 모듈은 'MC_getOemData()'를 통하여 상기 모바일 플랫폼으로 OEM 모듈이 제공하는 상기 플랫폼 내부에 저장된 데이터를 요청한다. 그리고 이에 대한 응답은 'ret: MC_getOemData:RET->SubEvt,Data'를 통하여 전달받는다.

그리고 상기 OEM 모듈의 blocking HAL은 다음과 같은 API 함수를 통하여 구현된다.

```
MH_gaInterface(M_Char* oemModule,
M_Int32 modInstanceID, M_Char * cmd, void * param1,
M_Int32* param1Size, void* param2 , M_Int32* param2Size
) {
    IF(oemModule == OEMMOD1)
    {
        SWITCH CASE(CMD)
        CASE CMD1:
            Return MH_B_1(modHandlerID,PARAM1,PARAM2);
            // CMD1으로 정의한 OEM MODULE 의 HAL A 를 호출
        CASE CMD2 :
            return MH_B_2(modHandlerID,PARAM1,PARAM2);
            // CMD2로 정의한 OEM MODULE 의 HAL B 를 호출
        .....
        DEFAULT : return M_E_NOTSUP;
```

```

}

ELSE IF(oemModule == OEMMOD2)

{

...

}

ELSE

return M_E_NOTSUP;

}

```

여기서, 상기 밑줄 그은 'MH_B_1(modHandlerID.PARAM1.PARAM2);'와 'MH_B_2(modHandlerID.PARAM1.PARAM2);'는 OEM 모듈 별로 해당 HAL을 정의하는 함수로서 MH_B_1 및 MH_B-2은 각각 다음과 같다.

```

[MH_B_1]

.....

Param1=ret1

Param2=ret2

[MH_B_2]

.....

Param1=ret1,

Param2=re2.....

```

예를 들어 OEM 모듈 1에서 제공하는 blocking HAL 'MH_B_1'은 다음과 같이 정의된 것이다.

```

MH_B_1()

{

// blocking 모드로 동작 한다.

// cmd1 에 해당하는 기능을 제공 한다.

//OEM 모듈의 인스턴스 관리는 hal 에서 관리 한다.

//파라미터1,2 의 정의에 따라 리턴값을 넘겨 준다.

}

```

한편, 상기 OEM 모듈의 기능이 non-blocking 모드에서 동작하는 경우(S500) 또는 Async 모드로 동작하는 경우가 있다. 이때 정상적인 경우 OEM 모듈에서는 M_E_WOULDBLOCK을 반환한다.

상기 OEM 모듈의 기능이 non-blocking 모드에서 동작하는 경우도 상기 blocking 모드와 마찬가지로 터널링 기능에서 제공하는 이벤트를 통하여 상기 데이터의 전달이 이루어진다. 먼저 상기 어플리케이션 모듈이 'MC_gaInterface(): CMD->GetData'를 통하여 상기 모바일 플랫폼으로 데이터를 요청한다. 상기 모바일 플랫폼은 보안 레벨의 체크 후 'MH_gaInterface(): CMD->GetData'를 통하여 상기 OEM 모듈로 데이터를 요청한다. 상기 모바일 플랫폼은 상기 OEM 모듈로부터 'ret: MH_gaInterface():ret->True'를 통하여 상기 요청에 대한 응답을 전달받는다. 그리고 상기 전달받은 응답은 'ret: MC_gaInterface():ret->True'를 통하여 상기 어플리케이션 모듈로 전달된다. 그리고 상기 요청한 데이터는 OEM 모듈의 HAL을 통하여 이벤트로 전달한다. 상기 이벤트는 OEM 모듈로부터 모바일 플랫폼으로 전달되는 'MH_pltSendEvent(): MH_KTFUSEROEM_EVENT->SubEvt'와 상기 모바일 플랫폼 내부에 이벤트 큐를 저장하고 상기 어플리케이션 모듈로 전달되는 'MH_KTFUSEROEM_EVENT'를 사용한다. 그리고 상기 어플리케이션 모듈은 'MC_getOemData()'를 통하여 상기 모바일 플랫폼으로 OEM 모듈이 제공하는 상기 플랫폼 내부에 저장된 데이터를 요청한다. 그리고 이에 대한 응답은 'ret: MC_getOemData:RET->SubEvt'를 통하여 전달받는다.

그리고 상기 OEM 모듈의 non-blocking HAL은 다음과 같은 API 함수를 통하여 구현된다.

```

MH_gaInterface()

{

IF(oemMoule == OEMMOD1)

{

SWITCH CASE(CMD)

CASE CMD1:

MH_A_1(modHandlerID,PARAM1,PARAM2);

Retrun M_E_WOULDBLOCK

// CMD1으로 정의한 OEM MODULE 의 HAL A 를 호출

// ASYNC 한 함수이므로 wouldblock 을 리턴한다.

CASE CMD2 :

return MH_B_2(modHandlerID,PARAM1,PARAM2);

// CMD2로 정의한 OEM MODULE 의 HAL B 를 호출

.....

DEFAULT : return M_E_NOTSUP;

}

ELSE IF(oemmoule == OEMMOD2)

{

```

```
...
```

```
}
```

```
ELSE
```

```
return M_E_NOTSUP;
```

```
}
```

여기서, 상기 밑줄 그은 'MH_A_1(modHandlerID,PARAM1,PARAM2);'와 'MH_B_2(modHandlerID,PARAM1,PARAM2);'는 OEM 모듈 별로 해당 HAL을 정의하는 함수로서 MH_A_1 및 MH_B-2은 각각 다음과 같다.

```
[MH_A_1]
```

```
.....
```

```
MH_PltSendEvent
```

```
(MH_KTFUSEROE
```

```
M_EVENT,data ...)
```

```
[MH_B_2]
```

```
.....
```

```
Param1=ret1
```

```
Param2=ret2
```

예를 들어, OEM 모듈 1에서 제공하는 non-blocking HAL 'MH__1'은 다음과 같이 정의된 것이다.

```
MH_A_1()
```

```
{
```

```
// NON-Blocking 모드로 동작 한다.
```

```
// cmd 에 해당하는 기능을 제공 한다.
```

```
//OEM 모듈의 인스턴스 관리는 hal 에서 관리 한다.
```

```
// 플랫폼에 Callback 이벤트를 전달 후 플랫폼에서 blocking 함수로 데이터를 가져 오게 한다.
```

```
}
```

한편, 상술한 바와 같은 절차를 통하여 이루어지는 상기 터널링 기능을 통한 데이터 전달 단계(S230, S240)에서 상기 데이터는 터널링 기능이 정의된 모바일 플랫폼에 이벤트를 통하여 전달한다.

이때, 모바일 플랫폼에 상술한 바와 같이 blocking mode와 non-blocking mode별로 새로이 정의된 HAL을 이용하여 OEM 모듈에서 모바일 플랫폼으로 데이터를 전달할 수 있다.

이때 사용되는 함수 및 파라미터는 다음과 같다.

[함수]

MH_pltSendEvent(MH_KTFUSEROEM_EVENT, &MH_KTFUserOEMEvent);

MH_pltSendEvent(MH_KTFUSEROEM_LONG_EVENT, &MH_KTFUserOEMLongEventCB);

표1. [파라미터]

이벤트	매개 변수
MH_KTFUSEROEM_EVENT	MH_KTFUserOEMEvent의 포인터
MH_KTFUSEROEM_LONG_EVENT	MH_KTFUserOEMLongEventCB의 포인터

상기 'MH_KTFUSEROEM_EVENT'는 OEM 모듈에서 모바일 플랫폼에게 사용자 정의의 이벤트를 전달하고자 할 경우 해당 이벤트를 전달 한다. MH_KTFUSEROEM_LONG_EVENT와 달리 파라미터값에 상수값만 전달하며 모바일 플랫폼에서는 CALLBACK 형태로 데이터를 GET 할 필요가 없다.

여기서, 상기 이벤트를 전달하는 경우는 일반적인 이벤트를 전달하는 경우와 특정 OEM 모듈에서 이벤트를 전달하는 경우로 나누어질 수 있다.

먼저, 상기 일반적인 이벤트를 전달하는 경우는 특정 OEM 모듈에서 발생한 이벤트가 아닌 일반적인 단말 기능에 의해 생성되는 이벤트를 말한다. 이때, 상기 이벤트의 전달은 'MH_KTFUSEROEM_EVENT'의 SUB 이벤트를 정의하여 모바일 플랫폼에 전달한다. 상기 터널링 HAL에서 모바일 플랫폼으로 전달되는 이벤트 및 매개변수는 다음과 같다.

표2. 상기 터널링 HAL에서 전달되는 이벤트 및 매개 변수

이벤트	매개 변수
MH_KTFUSEROEM_EVENT	MH_KTFUserOEMEvent의 포인터

상기 'MH_KTFUSEROEM_Event' 이벤트를 모바일 플랫폼에 전달 하는 구조체는 다음과 같이 정의된다.

```
typedef struct _MH_KTFUserOEMEvent{
    M_Int32 Event; // 어플리케이션에게 전달할 SUB_EVENT
    M_Int32 prgID; // sub 이벤트를 전달할 어플리케이션 program id,
    M_Char* appID; // sub 이벤트를 전달할 어플리케이션 ID
    M_Int32 param1; // sub 이벤트를 통해 전달될 파라미터
} MH_KTFUserOEMEvent;
```

또한, 상기 MH_KTFUserOEMEvent의 구성은 다음과 같다.

표3. MH_KTFUserOEMEvent의 구성

이벤트	매개 변수
event	OEM이 어플리케이션에 전달 하는 SUB 이벤트 타입
prgID	OEM이 이벤트를 전달할 WPI 어플리케이션의 프로그램 id prgID가 -1이 아닌 경우 아래 appID는 무시한다.

appID	위의 prgID가 -1인 경우 appID에 해당하는 어플리케이션에 이벤트를 전달한다. prgID가 -1이고 appID가 NULL인 경우 Active된 어플에게 전달한다.
param1	SUB_EVENT 에 따른 어플리케이션에 전달할 파라미터1

상기 일반적인 이벤트를 전달하는 흐름은 먼저 상기 OEM 모듈에서 모바일 플랫폼으로 'MH_KTFUSEROEM_EVENT'를 전달한다. 상기 모바일 플랫폼은 내부에 이벤트 큐를 저장하고 상기 OEM 모듈로 'ret:MH_pltSendEvent'를 전달한다. 그리고 모바일 어플리케이션 'postEvent():MC_KTFUSEROEM_EVENT, subEvt,param1'을 전달한다.

다음에, 특정 OEM 모듈에서 모바일 어플리케이션에게 이벤트를 전달하는 경우는 특정 OEM 모듈에서 데이터를 모바일 플랫폼에 직접 또는 CALLBACK의 형식으로 데이터를 전달하고자 할 경우 해당 이벤트를 전달한다. 이때 사용되는 이벤트는 'MH_KTFUSEROEM_LONG_EVENT'의 SUB 이벤트로서 OEM 모듈과 어플리케이션 모듈에서 지정한 특정 CALLBACK 이벤트를 생성하여 모바일 플랫폼에 전달하면 모바일 플랫폼은 SUB 이벤트를 어플리케이션 모듈에게 전달된다. 상술한 바와 같은 이벤트 흐름은 설명하면 다음과 같다.

상기 'MH_KTFUSEROEM_LONG_EVENT'는 OEM 모듈에서 모바일 플랫폼에게 데이터와 이벤트를 동시에 전달 하고자 할 경우 해당 이벤트를 발생한다.

OEM 모듈에서 모바일 플랫폼으로 'MH_KTFUSEROEM_LONG_EVENT.MH_KTFUserOEMLongEventCB'를 전달한다. 상기 모바일 플랫폼은 내부에 이벤트 큐를 저장하고 상기 OEM 모듈로 'ret:MH_pltSendEvent()'를 전달한다. 그리고 상기 어플리케이션 모듈에게 'postEvent():KTFUSEROEM_LONG_EVENT'를 전달한다. 상기 모바일 플랫폼은 상기 어플리케이션 모듈로부터 'MC_getOemData()'를 전달받고 상기 어플리케이션 모듈로 'ret->MH_KTFUserOEMLongEventCB'를 전달한다.

이때, 상기 'KTFUSEROEM_LONG_EVENT' 이벤트를 모바일 플랫폼에 전달 하는 구조체는 다음과 같이 정의된다.

```
typedef struct _MH_KTFUserOEMLongEventCB {
    M_Int32 Event; // 어플리케이션에게 전달할 MH_SUB_KTFUSEROEM_EVENT
    M_Int32 prgID; // sub 이벤트를 전달할 어플리케이션 program id,
    M_Char* appID; // sub 이벤트를 전달할 어플리케이션 아이디
    M_Int32 moduleID; // event가 발생한 OEM 모듈 ID
    M_Int32 modInstanceID; //Event 발생한 OEM 모듈의 Instance ID
    void *param1; // sub 이벤트를 통해 전달될 파라미터
    M_Int32 param1Size;
} MH_KTFUserOEMLongEventCB;
```

또한, 상기 MH_KTFUserOEMLongEventCB의 구성은 다음과 같다.

표4. MH_KTFUserOEMLongEventCB의 구성

이벤트	매개 변수
EVENT	OEM 모듈이 어플리케이션에 전달 하는 SUB 이벤트 타입
PrgID	OEM 모듈이 이벤트를 전달할 WPI 어플리케이션의 프로그램 id. 플랫폼에서 호출한 HAL의 파라미터(Caller ID)를 할당한다. 플랫폼에서는 prgID가 -1이 아닌 경우 아래 appID는 무시한다.
AppID	위의 prgID가 -1인 경우 appID에 해당하는 어플리케이션에 이벤트를 전달한다. prgID가 -1이고 appID가 NULL인 경우 Active된 어플에게 전달한다.

Module ID	이벤트가 발생한 OEM Module ID
ModinstanceID	EVENT 가 발생한 oem 모듈의 Handler or instance ID(기본값 0)
Param1	Sub EVENT 에 따른 어플리케이션에 전달할 파라미터1
Param1 Size	파라미터1 데이터의 크기

한편, 상기 도면에서는 도시되지 않았으나 터널링 기능을 통하여 신규 탑재된 모바일 기능을 이용하는 것을 종료하고자 할 경우 각 OEM 모듈에 맞는 함수를 호출 한다. 이때 Instance가 지원되는 OEM 모듈은 해당 Instance를 종료 시킨다. 사용되는 API 함수는 다음과 같다.

[사용 함수]

M_Int32 MC_gaInterfaceL1~L7(

M_Int32 oemModule,

M_Int32 ModInstanceID,

M_ Char * cmd,

Void * param1,

M_Int32* param1Size

Void* param2 ,

M_Int32* param2Size

);

[파라미터]

[IN] oemModule= GA_KTF_MOD1, // OPEN 하고자 하는 OEM 모듈 ID

[IN] ModInstanceID=1, // Free 시킬 OEM 모듈의 InstanceID 값

//(instance를 지원 하지 않는 경우 0).

[IN] cmd = "GA_CLOSE_MOD", // OEM 모듈 OPEN 을 위한 커맨드

[IN] param1=NULL , // OPEN 된 OEM 모듈의 instance id 값이 리턴된다(1~n)

[IN] param1Size=0 // 버퍼의 사이즈

[IN] param2 =NULL, // 디폴트 처리

[IN] param2Size =0; // 디폴트

도 7은 본 발명의 실시예에 따른 신규 모바일 기능을 추가하는 방법을 적용한 모바일 장치의 구성도이다. 즉 신규 모바일 기능을 추가하는 방법을 적용한 모바일 장치(100)는 저장부(110), 제어 모듈(120) 및 기능 모듈(130)을 포함한다.

상기 저장부(110)는 제1 저장부(112), 제2 저장부(114) 및 제3 저장부(116)로 구성된다.

상기 제1 저장부(112)는 모바일 플랫폼이 제공하는 모바일 기능을 위한 함수들을 정의하고 프로토 타입을 제공하는 API 보관한다. 모바일 플랫폼에는 기본적으로 어플리케이션 모듈이 응용하기 위한 기본 API와 확장 API를 저장하는데 바로 이러한 API를 보관하는 저장부이다. 즉 모바일 플랫폼에 기본적으로 탑재된 모바일 기능을 정의하는 API를 보관한다.

상기 제2 저장부(114)는 상기 제1 저장부(112)에 보관된 API 이외에 추가된 API를 보관한다. 즉, 상기 제1 저장부(112)에 보관된 API가 정의하는 모바일 기능 이외에 추가된 신규 모바일 기능을 위한 함수의 정의 및 프로토 타입을 제공하는 API를 보관한다.

상기 제3 저장부(116)는 도 2 내지 도 6에서 설명한 터널링 기능을 정의한 터널링 API를 보관한다. 여기서 상기 터널링 API는 모바일 플랫폼에 영향을 주지 않고 상기 제2 저장부에 보관된 API에 의해 정의된 모바일 기능을 이용하기 위한 터널링 기능을 제공하는 함수들로 구성된다. 보다 상세하게는 상기 터널링 API는 어플리케이션 모듈이 상기 제2 저장부(114)에 보관된 API와 대응하는 기능 모듈(130)과 통신 하기 위한 구조, 상기 기능 모듈(130)이 제공하는 기능이 담긴 데이터를 전달받기 위한 이벤트 및 상기 이벤트를 상기 기능 모듈로부터 전달받는 구조를 정의한 함수로 구성될 수 있다. 또한, 상기 터널링 API는 상기 터널링 API를 이용할 수 있는 기능 모듈(130)의 종류를 정의하는 구조 및 상기 어플리케이션 모듈이 호출하고자 하는 기능 모듈이 정의된 것인지 여부를 판단하는 구조를 정의한 함수를 더 포함할 수 있다.

상기 제어 모듈(120)은 API 관리자 모듈(122) 및 터널링 관리자 모듈(124)로 구성된다. 상기 API 관리자 모듈(122)은 API의 기본적인 제어 기능을 담당하는 것으로서 제1 저장부(112) 및 제2 저장부(114)에 보관된 API의 다운로드/실행/저장/삭제 등을 관리한다. 그리고 상기 제1 저장부(112)에 저장된 API에 대해서 상기 API가 정의하는 기능을 구현하는 기능 모듈을 호출하고 이로부터 상기 기능 모듈이 제공하는 기능을 제공받는 일련의 동작을 담당한다.

상기 터널링 관리자 모듈(124)은 상기 제3 저장부에 저장된 터널링 API를 통한 터널링 기능을 수행하는 수단으로서 상기 제2 저장부(114)에 보관된 API와 이와 대응하는 기능 모듈에 대해 터널링 API가 정의하는 터널링 기능을 적용한다. 상기 터널링 관리자 모듈(124)은 세부적으로 정의부(126), 판단부(127), 호출부(128) 및 전달부(129)로 구성된다. 상기 정의부(126)는 상기 터널링 기능을 적용 가능한 모바일 기능을 정의하는 수단으로서 보다 상세하게는 상기 모바일 기능을 구현하는 기능 모듈(또는 OEM 모듈)의 종류를 모바일 플랫폼에 정의한다. 이때 상기 기능 모듈의 식별자를 이용한다.

상기 판단부(127)는 추가하고자 하는 모바일 기능이 상기 정의부(126)를 통하여 정의한 것인지 여부를 판단한다. 보다 상세하게는 상기 API 관리자 모듈(122) 제2 저장부(114)에 저장하고자 하는 API가 정의하는 모바일 기능이 터널링 기능의 적용 대상인지 여부를 판단한다. 즉 상기 모바일 기능을 구현하는 기능 모듈(130)의 식별자가 정의된 것인지 여부를 판단한다. 그리고 상기 판단 결과 상기 정의부(126)를 통하여 정의된 기능 모듈 및 이와 대응한 API 모듈의 추가를 결정한다.

상기 호출부(128)는 상기 판단부(127)의 추가 결정에 따라 추가된 상기 제2 저장부(124)에 저장된 API와 대응하는 기능 모듈을 호출한다. 이때 상기 제3 저장부(126)에 저장된 터널링 API를 이용한다.

상기 전달부(129)는 상기 호출부(128)가 호출한 기능 모듈이 제공하는 기능을 전달받는 수단이다. 이때 역시 상기 제3 저장부(126)에 저장된 터널링 API를 이용한다.

여기서, 상기 제3 저장부(116)에 저장된 터널링 API와, 상기 터널링 API를 이용하여 상기 정의부(126), 상기 판단부(127), 호출부(128) 및 전달부(129)의 동작은 구체적으로 도 6에서 설명한 내용을 참조한다.

한편, 이상에서 설명한 본 발명의 실시예는 장치 및 방법을 통해서만 구현이 되는 것은 아니며, 본 발명의 실시예의 구성에 대응하는 기능을 실현하는 프로그램 또는 그 프로그램이 기록된 컴퓨터로 판독가능한 기록 매체를 통해 구현될 수도 있으며, 이러한 구현은 앞서 설명한 실시예의 기재로부터 본 발명이 속하는 기술분야의 전문가라면 쉽게 구현할 수 있는 것이다.

그리고 본 발명의 권리범위는 상술한 실시예에 한정되는 것은 아니고 다음의 청구범위에서 정의하고 있는 본 발명의 기본 개념을 이용한 당업자의 여러 변형 및 개량 형태 또한 본 발명의 권리범위에 속하는 것이다.

발명의 효과

전술한 구성에 의하여, 터널링 기능을 이용하여 모바일 플랫폼의 수정없이 기존의 모바일 플랫폼이 지원하지 않는 새로운 기능을 추가할 수 있으므로 모바일 플랫폼의 기능 호환성 및 플랫폼 호환성을 제공한다. 또한 종래에 모바일 플랫폼 방식

에 비해 지속적인 기능 개선을 통한 서비스 연속성을 제공할 수 있다. 이로 인하여 신규 기능을 제공하는 서비스의 적용 시기를 단축하여 신규 서비스의 빠른 시장 대응력을 강화할 수 있다. 더불어 플랫폼 개발 비용을 절감하고 신규 플랫폼 포팅 비용을 절감할 수 있다.

도면의 간단한 설명

도 1은 종래에 신규 모바일 기능을 추가하는 개념도이다.

도 2는 본 발명에 따른 신규 모바일 기능을 추가하는 개념도이다.

도 3은 본 발명에 따른 터널링 기능을 보인 개념도이다.

도 4는 본 발명의 실시예에 따른 신규 모바일 기능을 추가하는 방법을 보인 순서도이다.

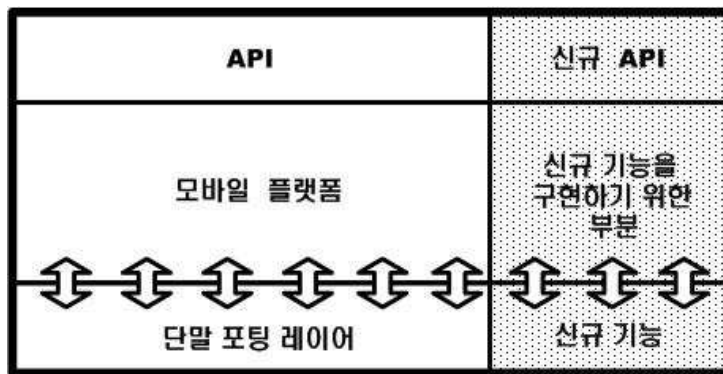
도 5는 본 발명의 또 다른 실시예에 따른 신규 모바일 기능을 추가하는 방법을 보인 순서도이다.

도 6은 본 발명의 또 다른 실시예에 따른 신규 모바일 기능을 추가하는 일련의 과정을 보인 흐름도이다.

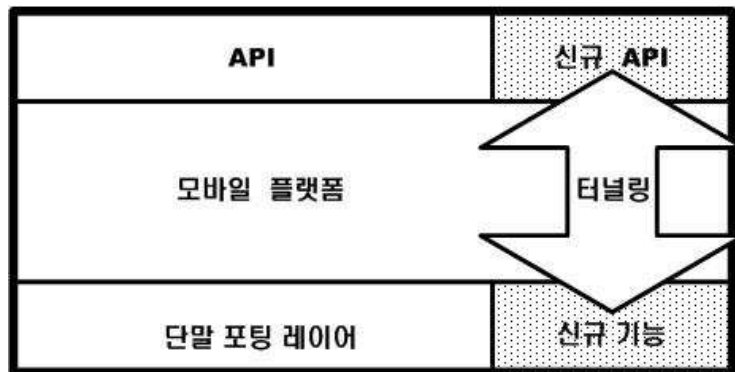
도 7은 본 발명의 실시예에 따른 신규 모바일 기능을 추가하는 방법을 적용한 모바일 장치의 구성도이다.

도면

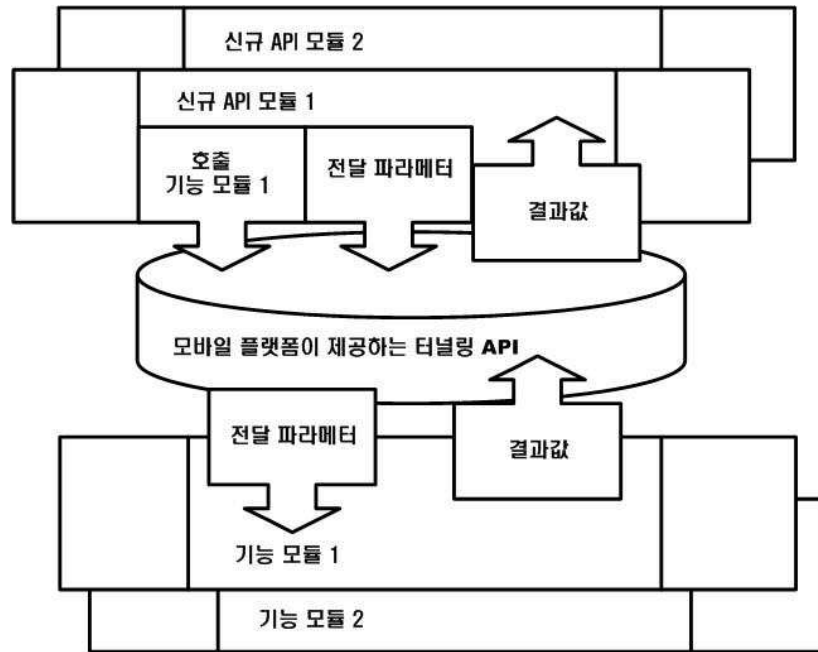
도면1



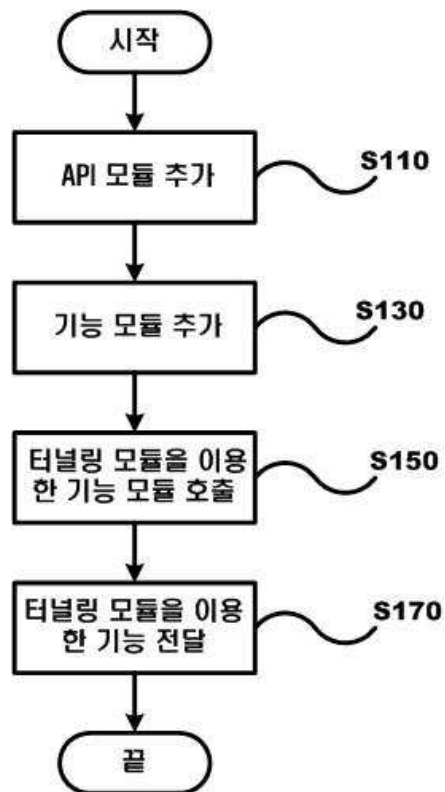
도면2



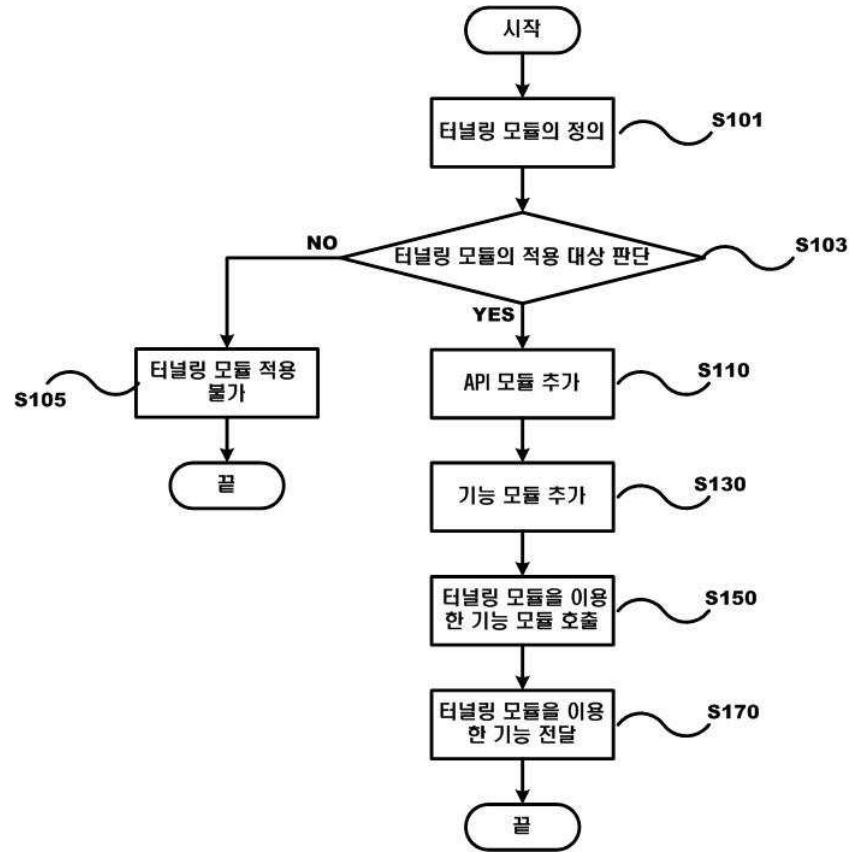
도면3



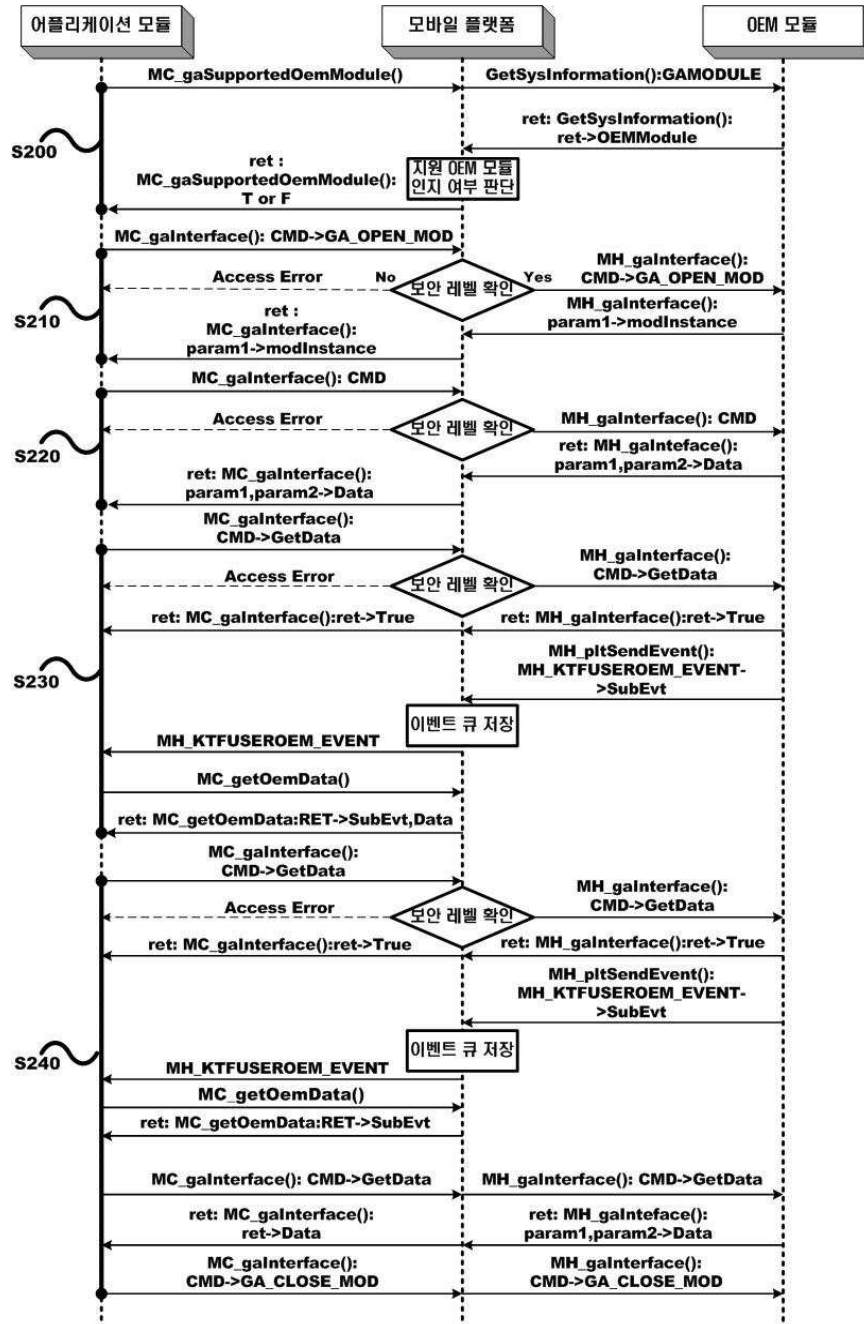
도면4



도면5



도면6



도면7

